DISS. ETH NO. 20729

TOWARDS ACCURATE STRUCTURED OUTPUT LEARNING AND PREDICTION

A dissertation submitted to
ETH ZURICH

for the degree of
DOCTOR OF SCIENCES

presented by
PATRICK ANDRÉ PLETSCHER
Master of Science in Computer Science (ETH Zurich)
born 2 January 1983
citizen of Marthalen, Switzerland

accepted on the recommendation of
Prof. Dr. Joachim M. Buhmann, examiner
Prof. Dr. Amir Globerson, co-examiner
Prof. Dr. Marc Pollefeys, co-examiner

2012

ABSTRACT

Structured output prediction is the task of learning and predicting with models where the hidden state of *several* random variables is described jointly. Recently these models became very popular as a growing number of applications have been identified to exhibit strong dependencies between the random variables describing the predicted output. In computer vision, which is the core application focus in this thesis, a structured model can for example incorporate prior assumptions about the smoothness of annotations of neighboring pixels. This thesis introduces *novel approximate methods* for the two core problems of structured models: *Parameter estimation* or *learning* and *prediction* or *inference*. Both tasks are intractable in the general case where the underlying (hyper)graph characterizing the dependencies between random variables contains a non-negligible number of cycles. Our learning and inference algorithms are based on two relatively simple, yet powerful ideas.

The first idea leverages on the fact that learning and prediction can be efficiently solved and are thus "easy", in models having an underlying *tree* or forest graph representation. We exploit this fact to devise approximate learning and inference algorithms in general graphs by decomposing the loopy graph into subsets of either nodes or edges with forest-like connectivity. Intelligent combinations of the computations carried out on the tree graphs, lead to accurate approximations of the original problem. In this thesis we demonstrate the efficacy of this approach when applied to both model parameter estimation, and obtaining the values of the variables minimizing the energy given by the model.

The second observation which sets the foundation for this work is that some optimization problems can be simplified by the introduction of a *smoothing parameter*, also referred to in the literature as a temperature. Using the smoothing approach we introduce a *unified model* for the two most popular discriminative structured models in use today, the Conditional Random Field and the structured Support Vector Machine. The unified model can lead to more accurate predictions. Furthermore, we demonstrate the value of the smoothing parameter in the context of

governing the *enforcement of non-convex constraints*. In particular, this approach is here applied for inferring the state with the minimum energy in structured models: Initially a convex linear programming relaxation is solved. By increasing the inverse temperature a non-convex quadratic programming relaxation arises, which however becomes easier to solve through the gradual decrease of the temperature. The resulting algorithm substantially outperforms state-of-the-art solvers based on linear programming relaxations at a minor increase in running time.

The described learning and inference approaches give rise to efficient and accurate algorithms for structured output prediction. In a related work, we also demonstrate that by directly modeling the problem at hand, the accuracy can be further increased: We characterize a sub-class of *high-order loss* functions for which the resulting learning problem is exactly solvable.

The methods studied in this thesis are applied to various problems in computer vision. For these applications the underlying graphical models are typically complex as many cycles are present. Furthermore, the training data sets are usually large and the base-line performance is often quite poor. More specifically, we study image denoising, cell detection in medical images as well as foreground-background segmentation.

ZUSAMMENFASSUNG

Structured output prediction beschreibt das Lernen und die Vorhersage mit Modellen in welchen die latenten Zustände von *mehreren* Zufallsvariablen gemeinsam beschrieben werden. In letzter Zeit sind solche Modelle sehr populär da eine wachsende Anzahl an Anwendungen identifiziert wurde welche starke Abhängigkeiten zwischen den Zufallsvariablen zeigen. Im Bildverstehen, welches die Haputanwendung dieser Arbeit ist, können strukturierte Modelle beispielsweise vorrangige Annahmen über die Glattheit von benachbarten Bildpunkten in einer Annotation berücksichtigen. Diese Arbeit präsentiert *neue approximative Methoden* für die zwei Kernprobleme von strukturierten Modellen: *Parameterschätzung* oder *Lernen* und die *Vorhersage* oder *Inferenz*. Beide Aufgaben sind im Allgemeinen schwierig zu lösen wenn die zugrundeliegenden (Hyper)graphen, welche die Abhängigkeiten zwischen Zufallsvariablen beschreiben, eine nicht vernachlässigbare Anzahl Zyklen aufweist. Unsere Lern- und Verhersagealgorithmen basieren auf zwei relativ simplen aber dennoch mächtigen Ideen.

Die erste Idee macht sich zu Nutzen, dass Lernen und die Vorhersage in Modellen wo die zugrundeliegende Repräsentation ein *Baum* oder Wald ist, effizient gelöst werden können und deshalb in einem gewissen Sinn "einfach" sind. Wir benutzen diese Tatsache um approximative Lern- und Vorhersagealgorithmen für allgemeine Graphen zu entwerfen. Die Algorithmen zerlegen den zyklischen Graphen in Untermengen von entweder Knoten oder Kanten welche eine waldförmige Konnektivität haben. Intelligente Kombinationen von Berechnungen auf den Bäumen führen zu akkuraten Approximationen für das ursprüngliche Problem. In dieser Arbeit demonstrieren wir die Wirksamkeit von diesem Vorgehen für die Parameterschätzung in strukturierten Modellen und die Berechnung des Zustandes, welcher eine gegebene Modellenergie minimiert.

Die zweite Beobachtung welche das Fundament für diese Arbeit bildet, besteht darin, dass gewisse Optimierungsprobleme durch das Einführen eines *Glättunsparameters*, auch Temperatur genannt, vereinfacht werden. Ein solcher Glättungsansatz erlaubt es uns ein *vereinheitlichtes Modell* für die zwei wichtigsten diskriminativen struktu-

rieren Modelle, das konditionierte Zufallsfeld und die strukturierte Stützpunktmaschine, einzuführen. Das vereinheitlichte Modell kann zu akkurateren Vorhersagen führen. Weiter demonstrieren wir den Nutzen des Glättungparameters im Zusammenhang mit dem *Erzwingen von nicht konvexen Bedingungen*. Im Besonderen wird dieser Ansatz hier für die Inferenz des minmalen Energiezustandes verwendet: Anfangs wird eine konvexe lineare Programm-Relaxierung gelöst. Durch das Erhöhen der inversen Temperatur entsteht ein nicht konvexes quadratisches Programm, welches aber einfacher zu lösen ist durch die graduelle Erhöhung der inversen Temperatur. Der resultierende Algorithmus übertrifft lineare Programm-Relaxierungen, welche auf dem neusten Stand der Technik sind, substantiell. Der Algorithmus hat nur eine geringe Erhöhung der Laufzeit zur Folge.

Die beschriebenen Lern- und Inferenzansätze führen zu effizienten und akkuraten Algorithmen für structured output prediction. In einer verwandten Arbeit zeigen wir auch, dass durch die direkte Modellierung des Problems die Genauigkeit weiter erhöht werden kann: Wir charakterisieren eine Unterklasse von *hochgradigen Verlustfunktionen* für welche das resultierende Lernproblem verlustfrei berechenbar ist.

Die Methoden welche wir in dieser Arbeit studieren, wenden wir auf verschiedene Probleme im Bildverstehen an. In diesen Anwendungen sind die zugrundeliegenden graphischen Modelle typischerweise komplex und weisen viele Zyklen auf. Weiter sind die Trainingsdatensätze oft gross und die Baseline Performanz ist häufig relativ schlecht. Im Speziellen studieren wir Bildwiederherstellung, Zelldetektion in medizinischen Bildern und die Vordergrund-Hintergrund Segmentierung.

PUBLICATIONS

The following publications are included in parts or in an extended version in this thesis:

- Patrick Pletscher, Cheng Soon Ong, and Joachim M. Buhmann (2009). „Spanning Tree Approximations for Conditional Random Fields." In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 5, pp. 408–415.
- Patrick Pletscher, Cheng Soon Ong, and Joachim M. Buhmann (2010). „Entropy and Margin Maximization for Structured Output Learning." In: *Proceedings of the 20th European Conference on Machine Learning (ECML)*. Vol. 6321. Lecture Notes in Computer Science, pp. 83–98.
- Patrick Pletscher et al. (2011). „Putting MAP back on the Map." In: *33rd Annual Symposium of the German Association for Pattern Recognition (DAGM)*. Vol. 6835. Lecture Notes in Computer Science. Springer, pp. 111–121.
- Patrick Pletscher and Sharon Wulff (2011). „A Combined LP and QP Relaxation for MAP." In: *NIPS Workshop on Discrete Optimization in Machine Learning (DISCML)*.
- Patrick Pletscher and Pushmeet Kohli (2012). „Learning low-order models for enforcing high-order statistics." In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 22, pp. 886–894.
- Patrick Pletscher and Cheng Soon Ong (2012). „Part & Clamp: An efficient algorithm for structured output learning." In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 22, pp. 877–885.
- Patrick Pletscher and Sharon Wulff (2012). „LPQP for MAP: Putting LP Solvers to Better Use." In: *Proceedings of the 29th International Conference on Machine Learning (ICML)*.

Furthermore, the following publications were part of my PhD research, are however not covered in this thesis. The topics of these publications are outside of the scope of the material covered here:

- Matthew Brand and Patrick Pletscher (2008). „A conditional random field for automatic photo editing." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Martin Jaggi et al. (2012). „Block-Coordinate Frank-Wolfe for Structural SVMs." In: *NIPS Workshop on Optimization for Machine Learning*.
- Simon Lacoste-Julien et al. (2013). „Block-Coordinate Frank-Wolfe Optimization for Structural SVMs." In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. To appear.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor, Joachim M. Buhmann, for advising me during my PhD studies. He gave me a lot of freedom to partially carry out my own research, witnessed through various collaborations. He has always been supportive in both, private and work related matters. Several discussions with Joachim sent me back to my desk to study various connections that he pointed out in more detail.

I would like to thank Amir Globerson and Marc Pollefeys for agreeing to review my thesis and their extremely valuable comments. It is an honor to have two such distinguished researchers on my PhD thesis committee.

I would also like to thank Cheng Soon Ong for numerous engaging research discussions, pointers to related material, critical questions or teaching me a relaxed attitude to research induced stress. Last but not least, Cheng also contributed a lot to a good social climate in our group, e.g. by introducing the Friday evening beer tradition or by extensive lunch discussions about practical work aspects, such as code management.

I was extremely fortunate to spend a summer at Microsoft Research in Cambridge, UK. I would like to thank Sebastian Nowozin, Pushmeet Kohli and Carsten Rother for making this a very insightful and productive internship. The diverse and broad skill set shared by this trio has influenced many aspects of my research.

Also, I would like to thank the members of the Machine Learning groups at ETH Zürich who made my PhD studies more fun. A big "thank you" goes to the group's secretary, Rita Klute, who made sure that everything is running smoothly.

Furthermore, I enjoyed interacting with the following people at conferences, at MSRC or on other occasions: Danny Tarlow, Inmar Givoni, Ross Girshick, Martin Jaggi, Dhruv Batra, Tamir Hazan, Simon Lacoste-Julien, Mark Schmidt, Peter Orbanz and Andreas Krause. Some of the interactions were more scientific, some less.

In matters mostly unrelated to Machine Learning, I would like to express my gratitude to Michael, Cyril, Adi, Simon, Janick, Thomas,

their partners and many other friends. It is great to share important moments in my life with you, like celebrating my bachelor party in Barcelona or the actual wedding in Israel. But it is probably even more important to experience simple and more trivial things together, such as a cold beer at the lake of Zurich.

I was always encouraged by my parents and my brother. Sometimes they have also visited me during my stays abroad to weaken any signs of home sickness that I might suffer from (which admittedly were rather rare). Was great fun to watch the 2010 soccer world cup final with my little brother in Cambridge.

Finally, I would like to express my outmost gratitude to Sharon Wulff. Over these past years you've become my hiking and skiing buddy, manuscript proof-reader, collaborator and most importantly life partner and wife. Thank you for all your support!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

This chapter introduces the basic setting of discriminative structured output prediction and details our contributions. We conduct a high-level discussion and defer to Chapter 2 for a more formal introduction to the topic.

1.1 STRUCTURED OUTPUT PREDICTION

A structured model is broadly speaking, a model in which *dependencies* among random variables are present. Prediction and parameter estimation in this model therefore need to reason about some variables jointly, as opposed to considering them independently. These dependencies generally increase the modeling power, and thus the accuracy of the model. Consider for example the task of predicting a part-of-speech tag sequence for every word in a sentence. In a simplified form we can for example imagine that the set of possible tags consists of a noun, verb, determiner (dt), preposition (prep) and adjective (adj). The tags of two neighboring words are clearly dependent, as a noun often follows an adjective etc., see Figure 1.1 for an illustration. Such dependencies could not be expressed in an independent model.

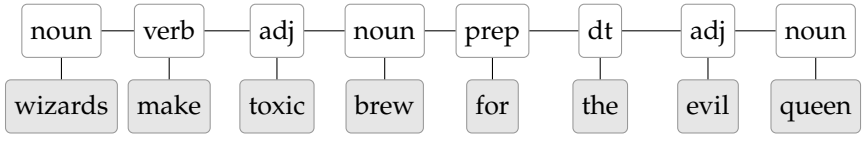


Figure 1.1: Simplified part-of-speech tagging example. The shaded observed words are labeled with different part-of-speech tags. The model assumes a chain structure for the dependencies between the word labels.

Another example is foreground-background segmentation, discussed in more detail in Chapter 4. Given an image showing an object, such as the scissors in Figure 1.2, we would like to predict the location of the object. More specifically, we aim at predicting for each pixel, whether

it is part of the foreground, i.e. the object itself, or the background. Again, as for the part-of-speech tagging example, one would expect

(a) Input image (b) Segmentation

Figure 1.2: The input image on the left is segmented into foreground and background pixels to the right. Foreground pixels are shown in white and background pixels in black. The data is taken from the GrabCut dataset.

that the label (i.e., background or foreground) of a pixel is dependent on closeby pixels. If we know that all of the neighbors of a particular pixel are labeled background, then the pixel itself is most likely also labeled background. Exactly this type of reasoning lies at the heart of structured models. However, the increased modeling power also comes at an increase in both statistical and computational complexity when compared to independent models. Structured models in which the underlying dependencies are specified by a loop-free graph, such as the part-of-speech example, are generally efficiently learnable. For more complex dependencies this is no longer the case and one has to resort to approximate approaches. Approximate methods for learning and prediction in structured models are the focus of this thesis. We explore different trade-offs in terms of accuracy and efficiency and also characterize particular settings in which learning and prediction are tractable.

Structured models root in Ising's seminal work in the 1920s (Ising 1925). The models were later generalized and many basic connections and techniques have been established in the 1970s and 1980s (Hammersley and Clifford 1971; Besag 1974; Geman and Geman 1984). These approaches, also known as Markov Random Fields (MRFs), have been fairly popular in the computer vision research community. MRFs received a renewed interest by the machine learning community with the introduc-

tion of the Conditional Random Field (CRF) (Lafferty, McCallum, and Pereira 2001) and the structured support vector machine (Tsochantaridis et al. 2005; Taskar, Guestrin, and Koller 2003). These modern structured models are in most aspects almost identical to traditional MRFs, but differ in one important aspect: the models are discriminative, which means they *condition on the observed data*. This enables the use of almost arbitrarily complex features, with usually only a minor effect on the runtime of the resulting structured classifiers. For many applications these more complex feature dependencies were an enabling technique to improved prediction accuracy (Mccallum, Freitag, and Pereira 2000). It turned out that practitioners, such as the computer vision community, have already successfully applied these discriminative structured models, long before they were introduced. However, most often the parameters of the models were (and still are) manually optimized. While this approach works reasonably well for small parameter spaces, for complicated high-dimensional scenarios, it is bound to fail. The CRF and the structured support vector machine are two examples of structured models, that also formulate the *learning* of these model parameters in a principled way.

1.2 CONTRIBUTIONS

The core problem that we study in this thesis is learning and prediction in intractable structured models. While the aforementioned part-of-speech tagging application has a linear dependency structure for which dynamic programming approaches lead to efficient computations for finding the most probable labeling, no such algorithms are known for general cyclic dependencies. Such loopy graphs are commonly encountered in computer vision applications, such as the foreground-background segmentation task briefly sketched above. Our work introduces an efficient message-passing algorithm for computing the most probable prediction in a general pairwise graphical model. The algorithm improves on the widely adopted linear programming relaxation. Further, we derive an extension of composite likelihood for learning the parameters of these intractable structured models.

We demonstrate in a second related line of research that the accuracy of these structured classifiers can be further improved even in tractable settings. Our contributions are two fold: First, we introduce a novel surrogate loss, which can lead to increased accuracy in scenarios with

extensive class overlap. Second, a novel learning algorithm for exact high-order loss minimization is given, enabling an improved modeling of some labeling problems encountered in medical imaging.

More specifically, the contributions of this thesis can be summarized as follows.

1. *We show that a combination of a linear programming relaxation and a quadratic programming relaxation can lead to improved algorithms for Maximum-A-Posteriori (MAP) inference (Chapter 3).* The inferred labelings show lower energies than the ones obtained using linear programming based algorithms at a minor increase in running time. The combined formulation is based on Kullback-Leibler penalty terms that force the linear and quadratic programming solutions to agree. The resulting optimization problem is efficiently solved using a message-passing algorithm.

2. *We characterize a set of high-order loss functions for which learning in a structured support vector machine is tractable (Chapter 4).* We study in detail the count-loss for binary segmentation which considers the absolute difference between the number of foreground pixels in a prediction and the ground-truth segmentation.

3. *We give a detailed theoretical analysis and empirical comparison of the two dominant structured output learning approaches (Chapter 5).* We introduce a unifying surrogate loss for structured output learning. The surrogate loss is parameterized by an additional hyperparameter. Extremal points of this hyperparameter correspond to the CRF and the structured support vector machine.

4. *We introduce a novel structured lower bound for the training of CRFs (Chapter 6).* The lower bound generalizes the composite likelihood and allows to balance the computational efficiency of learning versus the accuracy of the parameter estimation.

The thesis is organized as follows: First we give an in-depth introduction to the topic (Chapter 2). The main content of the thesis in the first part studies score maximizing models (Chapter 3 and Chapter 4), followed by probabilistic approaches (Chapter 5 and Chapter 6) in the second part. We conclude by giving an outlook on promising extensions of our work in Chapter 7.

1.3 A REMARK REGARDING COMPUTATIONAL COMPLEXITY

Throughout this thesis we will assume that $P \neq NP$. Expressions such as "intractable" or "exponentially large" should always be understood with respect to this assumption. We will however not state this assumption every time explicitly.

2

BACKGROUND

This chapter introduces our notation and the fundamental concepts underlying structured output learning and prediction. Recent text books on the topic such as (Koller and Friedman 2009), (Bakir et al. 2007) or (Nowozin and Lampert 2011) cover many of the concepts in similar depth.

input and output variables

In structured output prediction, the task is predicting interdependent output variables $\boldsymbol{y} \in \mathcal{Y}$[1] for a given input variable $\boldsymbol{x} \in \mathcal{X}$. An individual output variable y_i has the *discrete and finite* output domain $\mathcal{Y}_i$. For example for the task of foreground-background segmentation mentioned in Chapter 1, the output variables $\boldsymbol{y}$ correspond to the full segmentation of the image and y_i denotes the assignment of the i-th pixel to either foreground ($y_i = 1$) or background ($y_i = 0$). The output domain $\mathcal{Y}$ is hence of size 2^M, where M denotes the number of pixels of the image. Finally, $\boldsymbol{x}$ corresponds to the image for which a segmentation is computed. The core problem in structured output prediction arises from the combinatorial explosion also present in the foreground-background segmentation task. An exponential number of possible outputs have to be evaluated in order to choose the one that minimizes some cost function. The problem becomes even harder when the cost function itself is learned from training examples, as a modified prediction step has to be performed for different cost functions.

The remainder of this chapter is organized as follows: Section 2.1 discusses the optimal prediction according to Bayesian decision theory, Section 2.2 introduces factorized energy models defined by a factor graph or an undirected graphical model. Section 2.3 considers a dis-

1 Note that in our notation we assume that all inputs have the same output domain $\mathcal{Y}$. This is generally not fulfilled in practice, as e.g. a larger image obviously has a larger segmentation output domain than a smaller image. The notation would become less readable in case we would incorporate this possibility. In practice this however does not pose a problem and in our experiments we will also consider applications where the output domain is different for different examples.

criminative modelling approach for structured domains, often referred to as Conditional Random Field (CRF). Section 2.4 introduces the general theory of exponential family distributions, which is an essential tool for the understanding of many aspects of CRF models. Section 2.5 covers parameter estimation in structured models, of which several applications are sketched in Section 2.6. Finally, Section 2.7 and Section 2.8 discuss exact and approximate inference for structured models and its applications to learning the model parameters.

2.1 LOSS FUNCTIONS AND BAYESIAN DECISION THEORY

This section addresses the following question: "For a given posterior distribution $P(\boldsymbol{y}|\boldsymbol{x})$, what is the optimal way to predict output variables $\boldsymbol{y}$ for a given input variable $\boldsymbol{x}$?" For now we treat $P(\boldsymbol{y}|\boldsymbol{x})$ as a black-box that we assume to be known, the discussion in the next sections will cover the estimation of this posterior distribution from labeled training data.

loss function

In order to answer the question of optimal prediction, one has to first define a *loss function* $\Delta_{\boldsymbol{y}^*}(\boldsymbol{y})$ measuring the error of predicting $\boldsymbol{y}$ when the ground-truth is $\boldsymbol{y}^*$. We assume the following properties of a loss function:

1. Non-negativity: $\Delta_{\boldsymbol{y}^*}(\boldsymbol{y}) \geq 0 \quad \forall \boldsymbol{y}$.
2. Zero for the ground-truth: $\Delta_{\boldsymbol{y}^*}(\boldsymbol{y}^*) = 0$.
3. Upper bounded by one, i.e. $\Delta_{\boldsymbol{y}^*}(\boldsymbol{y}) \leq 1 \quad \forall \boldsymbol{y}, \boldsymbol{y}^*$.

While the last condition is not strictly necessary, we shall make this assumption throughout the thesis as it provides a range restriction on the loss. The following are popular and simple choices for loss functions:

ZERO-ONE LOSS Error of zero if the correct output is predicted and one otherwise. Formally the loss reads as follows

$$\Delta^{\text{zero-one}}_{\boldsymbol{y}^*}(\boldsymbol{y}) := 1 - \mathbb{I}_{\boldsymbol{y}^*}(\boldsymbol{y}).$$

Here $\mathbb{I}_{\boldsymbol{y}^*}(\boldsymbol{y})$ is a multivariate version of the indicator function, which is zero if all the elements of $\boldsymbol{y}$ agree with $\boldsymbol{y}^*$ and zero otherwise. While meaningful for multi-class classification settings

(assuming no similarity information between classes is available), for most scenarios involving several variables this loss is too strict: Surely, for the foreground-background segmentation example in the introduction, a segmentation that is only wrong in one of the pixels is preferable over one that gets all the pixels wrong. According to the zero-one loss both of these segmentations are equally bad and incur a loss of one.

HAMMING LOSS The Hamming loss, also referred to as per-variable zero-one loss, is a standard loss for sequences or other forms of segmentation. This loss favors predictions that lead to partially correct results, but does not assume any ordering on the label space $\mathcal{Y}_i$ of an individual output variable. The Hamming loss is defined as:

$$\Delta_{\boldsymbol{y}^*}^{\text{Hamming}}(\boldsymbol{y}) := \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \left[1 - \mathbb{I}_{y_i^*}(y_i) \right].$$

Here $\mathcal{V}$ denotes the set of all the output variables. The loss is normalized and hence, just like the zero-one loss, always confined to the range $[0, 1]$.

PER-VARIABLE SQUARED LOSS While the Hamming loss assumes a decomposition of the loss according to the different output variables y_i, it considers all the wrong predictions for variable y_i equally unfavorable. This weighting is relaxed by the squared loss, which penalizes wrong predictions by the squared distance between predictions y_i and y_i^*:

$$\Delta_{\boldsymbol{y}^*}^{\text{squared}}(\boldsymbol{y}) := \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \frac{1}{d_{max}(i)^2} \left(y_i^* - y_i \right)^2.$$

Here $d_{max}(i) = |\mathcal{Y}_i| - 1$ denotes the maximum possible difference between y_i and y_i^*. This term is needed in order to make the loss sum to at most one. For the task of image denoising, where an individual pixel y_i assumes gray-scale values in the range $y_i \in [0, 255]$, the squared loss is a much more meaningful loss than the Hamming loss: A gray-scale value difference of one between two pixels is clearly more favorable than a difference of 255, which would mean that the ground-truth pixel is white and the prediction is black, or vice versa.

It is straightforward to generalize the losses above to weighted versions, in which for each output variable y_i a weight $\alpha_i \in [0, 1]$ indicates its importance.

For a given loss and posterior $P(\boldsymbol{y}|\boldsymbol{x})$ one can derive the optimal predictor. According to Bayesian decision theory, minimization of the risk (or equivalently expected loss) leads to the best possible *prediction function* $f : \mathcal{X} \to \mathcal{Y}$ (Robert 2001, Theorem 2.3.2).

$$f^{\text{Bayes}}(\boldsymbol{x}) = \operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{Y}} \sum_{\boldsymbol{y}' \in \mathcal{Y}} \Delta_{\boldsymbol{y}'}(\boldsymbol{y}) P(\boldsymbol{y}'|\boldsymbol{x}). \tag{2.1}$$

minimum Bayes risk predictor

This prediction function is sometimes referred to as the *minimum Bayes risk* predictor as it yields to the optimal prediction function for a generic loss function Δ. However, the optimality of the prediction function is under the assumption that the true posterior $P(\boldsymbol{y}|\boldsymbol{x})$ is known. If this is not the case, optimality is no longer guaranteed. Below we discuss the resulting optimal prediction function for particular losses, an overview is given in Figure 2.1.

MAXIMUM-A-POSTERIORI MAP prediction is optimal for the zero-one loss and returns the label corresponding to the largest posterior probability:

$$f^{\text{MAP}}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y}' \in \mathcal{Y}} P(\boldsymbol{y}'|\boldsymbol{x}).$$

Efficient algorithms for computing the MAP prediction exist for many modeling assumptions, however these algorithms are often only approximate. Algorithms for MAP inference are described in detail in Section 2.7.

MAXIMUM POSTERIORI MARGINAL According to Bayesian decision theory, for the Hamming loss the Maximum Posteriori Marginal (MPM) is the optimal predictor, e.g. (Marroquin, Mitter, and Poggio 1987). It takes the following form

$$f^{\text{MPM}}(\boldsymbol{x}) = \operatorname*{argmax}_{y_i} P(y_i|\boldsymbol{x}) \quad \forall i.$$

Here $P(y_i|\boldsymbol{x}) = \sum_{\boldsymbol{y} \setminus y_i} P(\boldsymbol{y}|\boldsymbol{x})$ denotes the marginal of the i-th variable, obtained by integrating out all of the variables except for y_i. Marginals are often more difficult to compute than the MAP prediction, algorithms for this problem are again described in Section 2.7.

MINIMUM MEAN SQUARED ERROR The Minimum Mean Squared Error (MMSE), as the name indicates, is the best predictor for the squared loss. The output variables are predicted according to

$$f^{\text{MMSE}}(\boldsymbol{x}) = \mathbb{E}_{P(y_i|\boldsymbol{x})}[y_i] \quad \forall i.$$

The expectation is given as $\mathbb{E}_{P(y_i|\boldsymbol{x})}[y_i] = \sum_{y_i \in \mathcal{Y}_i} y_i P(y_i|\boldsymbol{x})$ and thus similarly to the MPM predictor, marginals need to be computed.

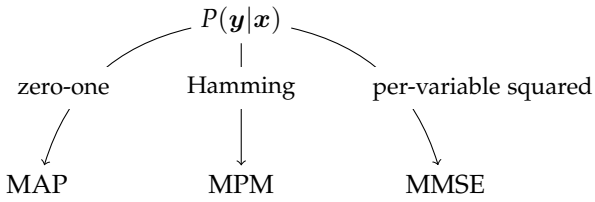


Figure 2.1: Optimal prediction approaches for different losses.

The three loss functions discussed here are rather simplistic. In fact, the Hamming loss and the squared loss both decompose into a sum over terms of the individual output variables, which simplifies some of the involved computations substantially. Often the loss used for the evaluation on real-world tasks, such as semantic segmentation, is more complex. Some examples of more complex losses are: the area under precision-recall curve, the F_1 score, and the peak signal-to-noise ratio. In Chapter 4 we discuss complex loss functions in more detail.

The current section investigated different loss functions and optimal prediction strategies. In the previous examples of prediction functions, we assumed that the *true* posterior $P(\boldsymbol{y}|\boldsymbol{x})$ is given. The next two sections will discuss modeling approaches that aim at accurately describing the true posteriori using parameterized factor graphs. These models include terms that allow the true data generating process to be modeled more accurately than in fully factorized models. However a gap between the true posterior and the model posterior still remains. We refer to this gap as a *misspecification*, that can potentially destroy some of the optimality guarantees given here.

2.2 FACTOR GRAPHS AND MARKOV RANDOM FIELDS

Factor graphs (Kschischang, Frey, and Loeliger 2001) are important for modeling complex systems of interdependent random variables. Their

applications are widespread and include computer vision, communication and signal or natural language processing. In this section we begin with an energy representation, the link to probabilistic models is made later through the Gibbs distribution. Factor graphs are a graphical notation for the joint distribution of random variables. A factor graph $\mathcal{FG} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$ contains two types of nodes: random variable nodes $\mathcal{V}$ and factor nodes $\mathcal{C}$. It is convenient to represent a factor graph as a diagram, where random variable nodes are illustrated by a circle ○. Direct dependencies between random variables are specified using factor nodes, shown as a rectangle ■. If a random variable is part of a dependency it is connected to the corresponding factor node. Therefore the resulting graph is a bipartite graph with the set of variable nodes on one side and the set of factor nodes on the other side. The energy

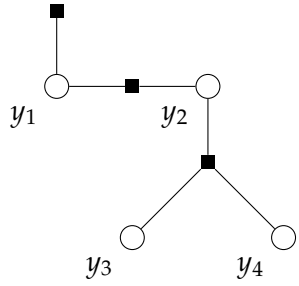
energy

$E(\boldsymbol{y})$ of a configuration $\boldsymbol{y}$ of random variables is then assumed to be given by

$$E(\boldsymbol{y}) = \sum_{c \in \mathcal{C}} \theta_c(\boldsymbol{y}_c).$$

potential

Here $\theta_c(\boldsymbol{y}_c) : \mathcal{Y}_c \to \mathbb{R}$ is referred to as a factor or potential and determines the energy contribution of the joint configuration $\boldsymbol{y}_c$. The subset notation restricts the variables to only those in factor c. Output configurations with smaller energies are favorable over those that have a higher energy. A large part of this thesis discusses approaches to learn suitable potentials from labeled training data, for now we however assume that they are given. Figure 2.2 shows an example of a simple factor graph. In the remainder we will use the convention of denoting



$$E(\boldsymbol{y}) = E(y_1, y_2, y_3, y_4) =$$
$$\theta_1(y_1) + \theta_{1,2}(y_1, y_2) + \theta_{2,3,4}(y_2, y_3, y_4)$$

Figure 2.2: A simple factor graph consisting of four variables. Its energy is written out to the right.

negative energies by $\bar{E}$, i.e. $\bar{E}(\boldsymbol{y}) := -E(\boldsymbol{y})$. $\bar{E}$ can also be thought of

score

as a score. Also, as sometimes instead of energy minimization we are

interested in score maximization we use the same convention for the factors $\bar{\theta}(\boldsymbol{y}_c) = -\theta(\boldsymbol{y}_c)$.

2.2.1 *The Gibbs Distribution*

Using the Gibbs distribution (also called the Boltzmann distribution) the energy $E(\boldsymbol{y})$ can be mapped to a distribution over output configurations $\boldsymbol{y}$:

$$P(\boldsymbol{y}) = \frac{1}{Z}\exp(\bar{E}(\boldsymbol{y})) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\exp(-\theta_c(\boldsymbol{y}_c)). \tag{2.2}$$

Here the *partition function* Z normalizes the distribution: *partition function*

$$Z = \sum_{\boldsymbol{y}\in\mathcal{Y}}\exp(\bar{E}(\boldsymbol{y})).$$

In the general case the computation of the partition function is computationally intractable (Valiant 1979), as it sums over the exponentially large set $\mathcal{Y}$. It is one of the main computational bottlenecks for the methods discussed in this thesis. Approximate evaluation of the partition function will be covered in more detail in Section 2.7. In the context *inverse temperature* of the Gibbs distribution it is common to include an additional inverse temperature parameter, β, into the exponent. The Gibbs distribution then becomes

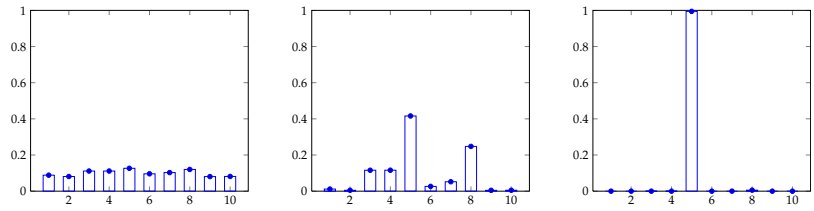
$$P_\beta(\boldsymbol{y}) = \frac{1}{Z(\beta)}\exp(\beta\bar{E}(\boldsymbol{y})). \tag{2.3}$$

The *inverse temperature* controls the "peakedness" of the distribution. For $\beta \to \infty$, $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$ concentrates on the MAP labels. For $\beta \to 0$, the posterior becomes a completely uniform distribution assigning probability $1/|\mathcal{Y}|$ to each output. This amplification behavior for increasing values of β is illustrated in Figure 2.3.

2.2.2 *Thermodynamic Potentials*

The Gibbs distribution has many important properties, a few of which are stated here. In the statistical physics literature some of these properties are introduced as relations between *thermodynamic potentials*, which are scalar functions of the inverse temperature β. The *free energy* is a scaled version of the logarithm of the partition function:

$$F(\beta) := -\frac{1}{\beta}\log Z(\beta).$$



(a) $\beta = 0.1$ (b) $\beta = 1$ (c) $\beta = 10$

Figure 2.3: Influence of the inverse temperature β on the Gibbs distribution over 10 outcomes.

The free energy is related to the *internal energy* and the *Shannon entropy* by the following identity

$$F(\beta) = \mathbb{E}_{P_\beta(\boldsymbol{y})}[E(\boldsymbol{y})] - \frac{1}{\beta} H(P_\beta). \tag{2.4}$$

Here the internal energy $\mathbb{E}_{P_\beta(\boldsymbol{y})}[E(\boldsymbol{y})]$ is the average of the energy

$$\mathbb{E}_{P_\beta(\boldsymbol{y})}[E(\boldsymbol{y})] := \sum_{\boldsymbol{y} \in \mathcal{Y}} P_\beta(\boldsymbol{y}) E(\boldsymbol{y}),$$

and the Shannon entropy is

$$H(P_\beta) := -\sum_{\boldsymbol{y} \in \mathcal{Y}} P_\beta(\boldsymbol{y}) \log P_\beta(\boldsymbol{y}).$$

The relation in (2.4) is central for many aspects of probabilistic inference and therefore the simple derivation is given below.

Proof.

$$\begin{aligned} F(\beta) &= \sum_{\boldsymbol{y} \in \mathcal{Y}} P_\beta(\boldsymbol{y}) E(\boldsymbol{y}) + \frac{1}{\beta} \sum_{\boldsymbol{y} \in \mathcal{Y}} P_\beta(\boldsymbol{y}) \log P_\beta(\boldsymbol{y}) \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}} P_\beta(\boldsymbol{y}) \left(E(\boldsymbol{y}) + \frac{1}{\beta} \log P_\beta(\boldsymbol{y}) \right) \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}} P_\beta(\boldsymbol{y}) \left(E(\boldsymbol{y}) - \frac{1}{\beta} \log Z(\beta) - E(\boldsymbol{y}) \right) \\ &= -\frac{1}{\beta} \log Z(\beta). \end{aligned}$$

Which proves the relation. □

We will revisit the identity in (2.4) in Section 2.4, where a variational formulation of the free energy is given in terms of a convex optimization problem over the probabilities $P_\beta(\boldsymbol{y})$.

2.2.3 *Pairwise Models*

For many applications the models studied contain only factors consisting of one or two output variables. The factor graph notation in this case is an unnecessary complication and hence in this setting the dependencies between random variables are often replaced by a standard graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The potentials $\theta_i(y_i)$ and $\theta_{ij}(y_i, y_j)$ are used to denote the cost of an assignment for a node $i \in \mathcal{V}$ and an edge $(i,j) \in \mathcal{E}$, respectively. The energy of such a pairwise model is then written as

$$E(\boldsymbol{y}) = \sum_{i \in \mathcal{V}} \theta_i(y_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(y_i, y_j). \tag{2.5}$$

Ising and Potts model

Figure 2.4 illustrates a model known as the *Ising model* in statistical physics (Ising 1925). On the left a factor graph representation is depicted

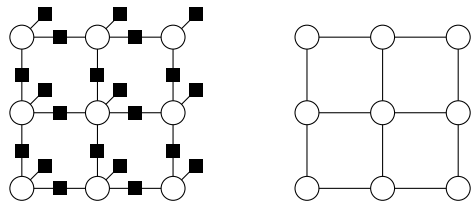


Figure 2.4: A graphical illustration of the Ising model. Lattice graphs are often also used in computer vision to balance a data fidelity term and the spatial smoothness between neighboring pixels. *Left*: The model expressed as a factor graph. *Right*: The same model as an undirected graphical model.

and on the right the model is shown as an undirected graphical model. The Ising model describes the joint configuration $\boldsymbol{y}$ of so called spins y_i which can be in two states $y_i \in \{-1, 1\}$. The underlying graph is usually assumed to be a lattice, in this thesis also called the 4-connected grid graph. The negative energy of the Ising model is then given by

$$\bar{E}(\boldsymbol{y}) = \sum_{i \in \mathcal{V}} y_i b_i + \sum_{(i,j) \in \mathcal{E}} a_{ij} y_i y_j. \tag{2.6}$$

Here a_{ij} denotes the interaction strength for edge (i, j) and b_i corresponds to an external field favoring either positive or negative spins for the i-th variable. Alternatively, the Ising model can also be written in the notation of (2.5) with $y_i \in \{0, 1\}$ (which is adopted in the remainder) by setting the unary potentials as $\theta_i(0) = b_i, \theta_i(1) = -b_i$ and the binary potential as $\theta_{ij}(0,0) = \theta_{ij}(1,1) = -a_{ij}$ and $\theta_{ij}(0,1) = \theta_{ij}(1,0) = a_{ij}$. This also paves the way for the extension of the Ising model to more than two states, known as the Potts model (Potts 1952). The potentials are called *attractive* if $a_{ij} > 0$, as the configuration in which neighboring variables take the same spin is preferred over those where the variables assume different labels. If $a_{ij} < 0$, the potentials are referred to as being *repulsive*. Figure 2.5 shows typical instances of the Ising model

attractive and repulsive potentials

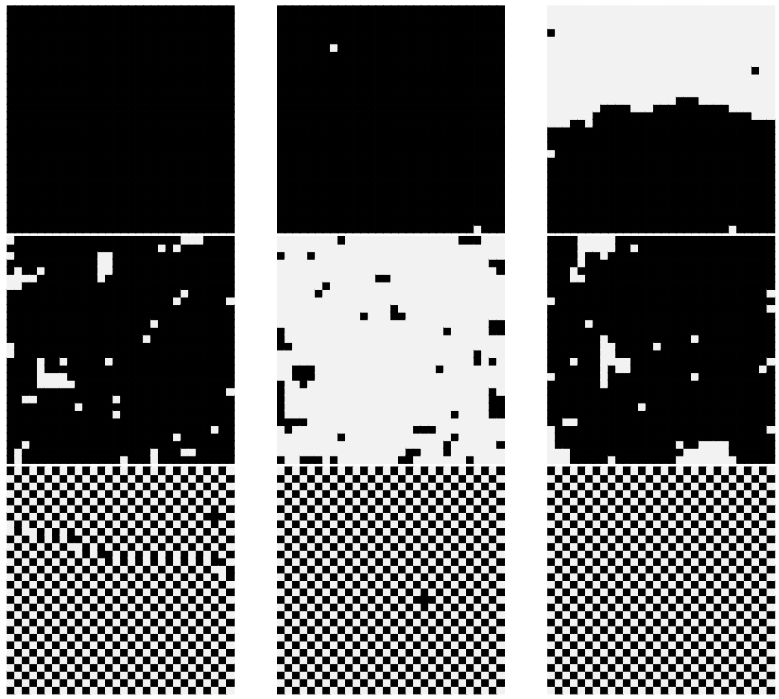


Figure 2.5: Samples from the Ising model (2.6) for $b_i \sim \text{Uniform}(-1, 1)$. a is chosen the same for all edges. The top row corresponds to $a = 1$, the middle row to $a = 0.5$ and the bottom row to a repulsive setting with $a = -1$. We used Gibbs sampling with 1000 sweeps, see Subsection 2.7.7.

for different choices of a and b. Most models studied in this thesis are

pairwise models, we therefore write the models in terms of the pairwise notation if possible.

2.2.4 *Markov Random Fields*

Factor graphs are a generalization of Markov Random Fields (MRFs). The literature on MRFs is extensive (Hammersley and Clifford 1971; Besag 1974; Geman and Geman 1984; Moussouris 1974). A full review of all the MRF properties is beyond the scope of this chapter. For a good overview, see for example (Winkler 2006, Chapter 3). Next we give a sketch of the main result, the Hammersley-Clifford theorem.

As the name implies, one of the key aspects of MRFs are characterizing *independencies*. Let us denote the neighbors of a variable i by $\mathcal{N}(i)$. For a pairwise undirected graphical model this set consists of all the nodes that share an edge with node i. For a factor graph the neighborhood is defined as all the variables that share a factor with node i. The set of neighboring variables is often called the *Markov blanket*. Let $\boldsymbol{y}_{\setminus i}$ denote the set of all variables except the i-th variable. The *local Markov property* states that for the conditional distribution of a variable given its neighborhood, the following equality holds:

Markov blanket

$$P(y_i|\boldsymbol{y}_{\setminus i}) = P(y_i|\boldsymbol{y}_{\mathcal{N}(i)}).$$

The Hammersley-Clifford theorem states that if a strictly positive distribution satisfies the local Markov property with respect to a graph G, then the distribution can be written as a MRF on G.

2.2.5 *High-order Potentials*

Traditionally, most work on MRFs and CRFs considered pairwise models. In the related field of Bayesian networks modeling, the factors studied often have larger cardinalities, though usually still a factor only depends on a small number of variables. Also, Bayesian networks often contain far fewer variables than the MRF models used for example in computer vision. The restriction to low-order models arises from two reasons. First, the introduction of high-order factors, i.e., factors that depend on many variables, comes at a massive increase in computational complexity due to the exponential explosion of the number of possible states of the factors. A second related problem is the representation of these factors: How can we efficiently describe the cost of

the different configurations without exhaustively listing all the energy configurations?

A recent line of research (Potetz 2007; Kohli, Kumar, and Torr 2009; Rother et al. 2009; Stobbe and Krause 2010; Tarlow, Givoni, and Zemel 2010) has studied these problems and characterized a number of high-order factors, e.g. factors that depend on all the variables in a model, for which one can efficiently compute the MAP configuration. One popular approach is to express a high-order factor in terms of an additional auxiliary variable and low-order factors (Kohli and Kumar 2010). In binary MRFs with submodularity assumptions, this approach, if applicable, often leads to an exact solution to the MAP problem. We will give more details about this characterization of high-order factors in Chapter 4. A second approach, leading to approximate MAP solutions, but more generic, is given in (Tarlow, Givoni, and Zemel 2010). The paper gives examples of high-order factors, for which the max-product messages can be computed efficiently. These messages are used in an approximate message-passing algorithm to infer the MAP labeling for a model containing such high-order factors. See Section 2.7 for more details about message-passing algorithms.

2.3 DISCRIMINATIVE MODELING

After having introduced graphical models and Bayesian decision theory, we can now turn to the main problem addressed in this thesis. *Structured output prediction* considers the task of predicting outputs $\boldsymbol{y}$ for given inputs $\boldsymbol{x}$. The previous section assumed that the energy function is known, this assumption is relaxed in the current section. We focus on *discriminative models*, which can be thought of as a prediction function $f : \mathcal{X} \to \mathcal{Y}$ as introduced in Section 2.1. It is important to point out that generative models, which model the joint distribution $P(\boldsymbol{y}, \boldsymbol{x})$ of input and output variables, are not studied here. Discriminative models do not impose any assumptions on the generation of $\boldsymbol{x}$, but only on the conditional distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ (or possibly unnormalized versions thereof). Therefore, we always make the assumption that the input variables $\boldsymbol{x}$ are fully observed. The parametric form of the prediction function, specified through a graphical model, is assumed to be given. The parameters $\boldsymbol{w}$ of this model, however, are assumed to be unknown and shall be learned from observed pairs of input and output variables. The graphical models therefore contain *two types of variables*,

the observed input variables and the unobserved output variables. Furthermore, the energy depends on the parameters $\boldsymbol{w}$ of the model. As we restrict ourselves to discriminative models, the dependency on $\boldsymbol{x}$ is computationally inexpensive, as it is observed, we therefore often do not specify the factorization on $\boldsymbol{x}$ explicitly. For the output variables on the other hand, the energy is assumed to factorize according to:

$$E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) = \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}(t)} \theta_c(\boldsymbol{y}_c, \boldsymbol{x}, \boldsymbol{w}_t) \tag{2.7}$$

Here we introduced another important concept, the *factor template* t, see also (Sutton and Mccallum 2012). $\mathcal{T}$ denotes the set of all differently parameterized factors. Each of its members t is a set of factors, sharing the same parameter $\boldsymbol{w}_t$.

To illustrate the concept of a factor template in more detail let us consider the simple Ising model again. We assume that for each variable $y_i \in \{-1, 1\}$ we observe a noisy measurement of the spin, denoted by $x_i \in [-1, 1]$. The energy of an input/output configuration is modeled as:

$$\bar{E}(\boldsymbol{y}, \boldsymbol{x}, w) = \sum_{i \in \mathcal{V}} x_i y_i + w \sum_{(i,j) \in \mathcal{E}} y_i y_j. \tag{2.8}$$

Here w denotes a scalar balancing the first and second term against each other. The first term can be understood as data-fidelity term and the second term as a smoothness term. A comparison to (2.7) shows that $\mathcal{T}$ simply consists of the unary and the pairwise potentials. The full factor graph is shown in Figure 2.6. In factor graphs and graphical models, observed variables (or input variables) are shown as a gray circle ◯. These variables do not pose any computational problems for inference, as they could alternatively always be included in the factors themselves. Also, when writing an energy in the undirected graphical model notation, we never include observed variables in the node set $\mathcal{V}$, despite the fact that we include these nodes for visualization in the graphical model. The factor template notation also allows for more complicated parametric forms, for example an Ising model where the horizontal and the vertical edges are parameterized differently

$$\bar{E}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) = \sum_{i \in \mathcal{V}} x_i y_i + w_h \sum_{(i,j) \in \mathcal{E}_h} y_i y_j + w_v \sum_{(i,j) \in \mathcal{E}_v} y_i y_j. \tag{2.9}$$

Here $\mathcal{E}_v$ denotes the vertical edges and $\mathcal{E}_h$ the horizontal edges.

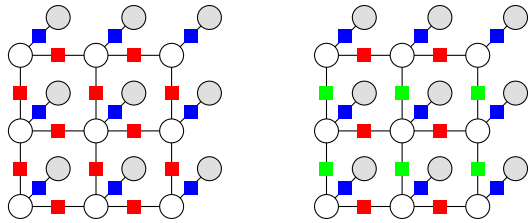


Figure 2.6: The Ising model with input and output variables. The color of a factor node indicates its factor template. *Left*: The model in (2.8). Here blue indicates the unary potentials and red the pairwise potentials. *Right*: The model in (2.9) where red indicates horizontal edge potentials and green the vertical potentials.

A common assumption (Koller and Friedman 2009, Section 4.4.1.2) is the linearity of the energy in the parameters. We shall also make this assumption throughout this thesis. The linear dependence simplifies the estimation of $\boldsymbol{w}$ considerably, as the most common estimation principles, such as maximum likelihood, are convex or concave optimization problems in this case. The parameterized negative energy of a factor graph as in (2.7) can therefore be rewritten as

$$\bar{E}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle = \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}(t)} \langle \boldsymbol{w}_t, \boldsymbol{\phi}_t(\boldsymbol{x}, \boldsymbol{y}_c, c) \rangle. \tag{2.10}$$

sufficient statistics

Here the *feature map* or *sufficient statistics* $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ of the model denotes a mapping from the input and output domain to a joint input/output space employed with an inner product. In this thesis we assume the joint feature map to be the Euclidean space, formally $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$. The mapping $\boldsymbol{\phi}$ is application-dependent and is implicitly specified by the graphical model and its parameterization. The right hand side of (2.10) makes use of the underlying factorization and rewrites the inner product in terms of parameters $\boldsymbol{w}_t$ for each factor template and corresponding feature maps $\boldsymbol{\phi}_t$. The feature map in this representation takes an additional input c the factor index, so that the feature extraction of $\boldsymbol{x}$ can be made factor dependent. Generally $\boldsymbol{w}_t$ resides in a lower-dimensional space than $\boldsymbol{w}$, unless there is only one factor template. One can either think of $\boldsymbol{w}_t$ as a lower dimensional vector and $\boldsymbol{w}$ as the concatenation of these lower dimensional vectors, or alternatively consider $\boldsymbol{w}_t$ to have the same dimension as $\boldsymbol{w}$, but spanning only a

subspace with all dimensions that are not associated with the t-th template set to zero. We adopt the latter view point as it simplifies the notation. We use $\hat{\boldsymbol{w}}_t$ and $\hat{\boldsymbol{\phi}}_t(\boldsymbol{x}, \boldsymbol{y}_c, c)$ to indicate that the dimensions not related to the t-th factor template were removed. Section 2.6 gives a number of examples of the sufficient statistics $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ for different tasks.

As before, the Gibbs distribution allows us to transform the energy of an output variable to the probability of an output variable. The important fact is that for a discriminative modeling approach one uses the *conditional* Gibbs distribution of an output variable given an input variable:

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{w})} \exp(\beta \bar{E}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})). \tag{2.11}$$

The difference between a conditional Gibbs distribution and a joint Gibbs distribution over $\boldsymbol{x}$ and $\boldsymbol{y}$ is only in the form of the partition function. For the joint distribution the partition function no longer depends on $\boldsymbol{x}$, as it is marginalized out. Therefore, depending on the exact dependence of the energy on $\boldsymbol{x}$, the computation of its corresponding partition function becomes much more difficult.

We will refer to a discriminative model of the form (2.11) as a Conditional Random Field (CRF) (Lafferty, McCallum, and Pereira 2001), even if the conditional probability in (2.11) is not an accurate estimation the true data generating posterior. Therefore the structured Support Vector Machine (SVM), which will be discussed in Subsection 2.5.2 is also considered a CRF model, simply with a different learning approach. The key difference of the CRF to the more classic MRF is the conditioning on the input variables $\boldsymbol{x}$, which in some applications lead to dramatic improvements in the prediction accuracy, as more complex and expressive features can be constructed (Lafferty, McCallum, and Pereira 2001; Mccallum, Freitag, and Pereira 2000).

2.4 EXPONENTIAL FAMILIES

This section introduces exponential family distributions and identifies the Gibbs distribution with the energy in (2.10) as a specific choice of an exponential family distribution. First, we however introduce a reparameterization of the energy. While the formulation of the energy in (2.10) is convenient for parameter estimation as it reveals the linear dependence on the parameters $\boldsymbol{w}$, for prediction an alternative, over-

complete representation is usually more insightful. Let use define the mapping $\boldsymbol{\varphi}(\boldsymbol{y}) : \mathcal{Y} \to \mathbb{R}^L$ which corresponds to a sufficient statistics of the output variables [2]. Using indicator variables each possible assignment for every factor in the factor graph is encoded. Hence, for a pairwise graphical model where each variable can assume K different values, $L = K|\mathcal{V}| + K^2|\mathcal{E}|$. The sufficient statistics corresponding to the nodes are given by

$$\varphi_{i;k}(\boldsymbol{y}) = \mathbb{I}_k(y_i),$$

and the ones corresponding to the edges by

$$\varphi_{ij;kl}(\boldsymbol{y}) = \mathbb{I}_{k,l}(y_i, y_j).$$

Here $\mathbb{I}_{k,l}(y_i, y_j)$ returns one when $y_i = k$ and $y_j = l$, and zero otherwise. These sufficient statistics are consequently concatenated to give $\boldsymbol{\varphi}(\boldsymbol{y})$. For models that include higher-order factors, similar statistics for these potentials need to be included. The full negative energy of an input/output configuration can then be written as a bilinear mapping in the parameters $\boldsymbol{w} \in \mathbb{R}^D$ and the overcomplete output variables feature map $\boldsymbol{\varphi}(\boldsymbol{y}) \in \mathbb{R}^L$:

$$\bar{E}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle = \boldsymbol{w}^\mathsf{T} \mathbf{Y}(\boldsymbol{x}) \boldsymbol{\varphi}(\boldsymbol{y}). \tag{2.12}$$

Here $\mathbf{Y}(\boldsymbol{x}) \in \mathbb{R}^{D \times L}$ copies the extracted features to the corresponding position according to $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$. Such a matrix $\mathbf{Y}(\boldsymbol{x})$ always exists. The joint input/output sufficient statistics $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ can thus be related to $\mathbf{Y}(\boldsymbol{x})$ and $\boldsymbol{\varphi}(\boldsymbol{y})$ by

$$\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{Y}(\boldsymbol{x}) \boldsymbol{\varphi}(\boldsymbol{y}).$$

Furthermore, it is often convenient to define a short-hand for the negative costs vector

$$\bar{\boldsymbol{\theta}}(\boldsymbol{x}) = \left(\boldsymbol{w}^\mathsf{T} \mathbf{Y}(\boldsymbol{x}) \right)^\mathsf{T}.$$

Again we use a bar to indicate that the interpretation is as a *negative energy*. For inference often the dependence of $\bar{\boldsymbol{\theta}}(\boldsymbol{x})$ on the input variables $\boldsymbol{x}$ is discarded, which we shall also do here. The formulation of the negative energy through the bilinear mapping in (2.12) allows us to include the conditioned variables $\boldsymbol{x}$ either in the cost parameters $\bar{\boldsymbol{\theta}}(\boldsymbol{x})$ or in the sufficient statistics $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$, depending on what is more

2 Note that the previously introduced sufficient statistics $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ is dependent on both the input and the output variables. Whereas $\boldsymbol{\varphi}(\boldsymbol{y})$ is only a sufficient statistics of the output variables.

convenient. Generally $D \ll L$ as otherwise learning $\boldsymbol{w}$ from data is an ill-specified problem, for example for the Ising model in (2.8), $D = 1$ whereas $L = 2|\mathcal{V}| + 4|\mathcal{E}|$.

We now turn to a discussion of *exponential family models*, sometimes also called *log-linear models* in the statistics literature. In our discussion we assume that observed variables $\boldsymbol{x}$ are already integrated into the parameters $\bar{\boldsymbol{\theta}}$ by the construction above. An exponential family model has the form

$$P(\boldsymbol{y}|\bar{\boldsymbol{\theta}}) = \exp(\langle \bar{\boldsymbol{\theta}}, \boldsymbol{\varphi}(\boldsymbol{y}) \rangle - A(\bar{\boldsymbol{\theta}})), \tag{2.13}$$

where the logarithm of the partition function is denoted by

$$A(\bar{\boldsymbol{\theta}}) := \log \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp(\langle \bar{\boldsymbol{\theta}}, \boldsymbol{\varphi}(\boldsymbol{y}) \rangle).$$

$\bar{\boldsymbol{\theta}}$ are then referred to as the canonical parameters of the distribution and are assumed to have some associated domain $\bar{\boldsymbol{\theta}} \in \Omega$. Often we will assume $\Omega = \mathbb{R}^L$, but sometimes certain modeling constraints will be expressed by restricting the domain Ω. The formulation of the exponential family model above is kept simple on purpose. Exponential family distributions are more general and for example also allow for continuous $\boldsymbol{y}$. For our discussion the form in (2.13) is however sufficient. Comparing (2.13) and (2.3) it is clear that the Gibbs distribution for an energy that is linear in $\boldsymbol{w}$, is also an exponential family model. Additionally, the same relation also holds for $\bar{\boldsymbol{\theta}}$ as the canoncial parameters. Note that the inverse temperature parameter β is here absorbed into $\bar{\boldsymbol{\theta}}$. The exponential family distributions have a number of desirable properties which are summarized below. The statements are given in terms of $\bar{\boldsymbol{\theta}}$ but also hold for $\boldsymbol{w}$ in slightly modified form. For proofs see for example (Wainwright and Jordan 2008, Proposition 3.1).

- The first two derivatives of the log partition function yield the moments of the random variable $\boldsymbol{\varphi}(\boldsymbol{y})$:

$$\frac{\partial A}{\partial \bar{\boldsymbol{\theta}}} = \mathbb{E}_{\bar{\boldsymbol{\theta}}}[\boldsymbol{\varphi}(\boldsymbol{y})] := \sum_{\boldsymbol{y}} P(\boldsymbol{y}|\bar{\boldsymbol{\theta}}) \boldsymbol{\varphi}(\boldsymbol{y}).$$

$$\frac{\partial^2 A}{\partial \bar{\boldsymbol{\theta}}^2} = \mathrm{Cov}_{\bar{\boldsymbol{\theta}}}[\boldsymbol{\varphi}(\boldsymbol{y})] := \mathbb{E}_{\bar{\boldsymbol{\theta}}}[\boldsymbol{\varphi}(\boldsymbol{y}) \boldsymbol{\varphi}(\boldsymbol{y})^\mathsf{T}] - \mathbb{E}_{\bar{\boldsymbol{\theta}}}[\boldsymbol{\varphi}(\boldsymbol{y})] \mathbb{E}_{\bar{\boldsymbol{\theta}}}[\boldsymbol{\varphi}(\boldsymbol{y})]^\mathsf{T}.$$

- $A(\bar{\boldsymbol{\theta}})$ is a convex function of $\bar{\boldsymbol{\theta}}$ on its domain Ω. This property follows directly from the positive semidefiniteness of the covariance matrix.

conjugate dual

Convex analysis provides us with an extremely powerful tool for studying $A(\bar{\boldsymbol{\theta}})$ through its *conjugate dual*, sometimes also called the Legendre-Fenchel transformation. The conjugate dual $A^*(\boldsymbol{\mu})$ of the log partition function is given by

$$A^*(\boldsymbol{\mu}) = \sup_{\bar{\boldsymbol{\theta}} \in \Omega} \{\langle \boldsymbol{\mu}, \bar{\boldsymbol{\theta}} \rangle - A(\bar{\boldsymbol{\theta}})\}. \tag{2.14}$$

marginal polytope

Notice that (2.14) is now a function of the dual variables $\boldsymbol{\mu}$. It will turn out that $\boldsymbol{\mu}$ has a particularly nice interpretation and coincides with the marginals (or sometimes also mean parameters) of $P(\boldsymbol{y}|\bar{\boldsymbol{\theta}})$. Let us introduce the constraint set $\mathcal{M}$, called the *marginal polytope* and defined as the marginals of the factors corresponding to a valid distribution $P(\boldsymbol{y}|\bar{\boldsymbol{\theta}})$:

$$\mathcal{M} = \{\boldsymbol{\mu} \mid \exists P(\boldsymbol{y}|\bar{\boldsymbol{\theta}}) \text{ with marginals } \boldsymbol{\mu}\}. \tag{2.15}$$

For a pairwise graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ the condition on valid marginals can be written as follows:

$$\left\{ \begin{array}{c} P(y_i) = \mu_i(y_i)\ \forall i \in \mathcal{V},\ \forall y_i \in \mathcal{Y}_i \\ P(y_i, y_j) = \mu_{ij}(y_i, y_j)\ \forall (i,j) \in \mathcal{E},\ \forall y_i \in \mathcal{Y}_i,\ \forall y_j \in \mathcal{Y}_j \end{array} \right\}.$$

For higher-order factors additional conditions would need to be included to ensure consistency for these marginals. The dimensionality of $\boldsymbol{\mu}$ is the same as $\bar{\boldsymbol{\theta}}$ and $\boldsymbol{\varphi}(\boldsymbol{y})$. As pointed out in (Wainwright and Jordan 2008, Section 3.4), (2.14) can also be understood as an alternative parametrization of the distribution $P(\boldsymbol{y}|\bar{\boldsymbol{\theta}})$ through the marginals instead of the canonical parameters $\bar{\boldsymbol{\theta}}$. A standard result recovers the negative Shannon entropy as the conjugate dual function of the log partition function, subject to the constraint that the marginals arise from a valid distribution:

$$A^*(\boldsymbol{\mu}) = \begin{cases} -H(P(\boldsymbol{y}|\bar{\boldsymbol{\theta}}(\boldsymbol{\mu}))) & \text{if } \boldsymbol{\mu} \in \mathcal{M} \\ +\infty & \text{otherwise.} \end{cases}$$

It is important to note that the entropy above is not explicitly formulated in terms of $\boldsymbol{\mu}$, but rather in terms of the distribution $P(\boldsymbol{y}|\bar{\boldsymbol{\theta}})$ associated with the marginals. We use $\bar{\boldsymbol{\theta}}(\boldsymbol{\mu})$ to denote the parameters that satisfy this relation:

$$\mathbb{E}_{\bar{\boldsymbol{\theta}}(\boldsymbol{\mu})}[\boldsymbol{\varphi}(\boldsymbol{y})] = \nabla A(\bar{\boldsymbol{\theta}}(\boldsymbol{\mu})) = \boldsymbol{\mu}.$$

Using convex analysis the log partition function can be rewritten as a convex optimization problem over the marginal polytope $\mathcal{M}$:

$$\begin{aligned} A(\bar{\boldsymbol{\theta}}) &= \sup_{\boldsymbol{\mu} \in \mathcal{M}} \{\langle \bar{\boldsymbol{\theta}}, \boldsymbol{\mu} \rangle - A^*(\boldsymbol{\mu})\} \\ &= \sup_{\boldsymbol{\mu} \in \mathcal{M}} \{\langle \bar{\boldsymbol{\theta}}, \boldsymbol{\mu} \rangle + H(P(\boldsymbol{y}|\bar{\boldsymbol{\theta}}(\boldsymbol{\mu})))\}. \end{aligned} \tag{2.16}$$

A similar formulation was given in (2.4), the key difference here is that it recasts the problem of computing the marginals and the partition function as a convex optimization problem. There are two major problems with the optimization problem in (2.16). First, the constraint set $\mathcal{M}$ is complicated (exponential in size). Second, the Shannon entropy is generally not explicitly expressible in $\boldsymbol{\mu}$, but rather implicitly in the distribution $P(\boldsymbol{y}|\bar{\boldsymbol{\theta}})$ associated with the marginal vector. Hence, one can in general not rewrite the entropy as a sum of unary and pairwise entropy terms, which would simplify the problem substantially. For the special case of a tree, such a factorization of the entropy however exists, see Subsection 2.7.2. Approximations of the constraint set and the entropy lead to efficient variational inference approaches which are discussed in Subsection 2.7.4 and Subsection 2.7.5. Finally, to give an intuition about conjugate duality, Figure 2.7 illustrates how the function $\log(x)$ can be reexpressed as a maximization problem over simple linear functions.

2.5 LEARNING

We study the parameter estimation problem, or equivalently learning, from the point of *Empirical Risk Minimization (ERM)* (Vapnik 1995). ERM is defined w.r.t. a given data set $\{(\boldsymbol{x}^n, \boldsymbol{y}^n)\}_{n=1}^N$[3] and a loss function $\ell(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$. Often the loss is a convex surrogate function of the true loss function Δ introduced in the section on Bayesian decision theory. The need for replacing the true loss by a surrogate convex loss arises from the fact that the true loss is difficult to optimize for as it leads to a non-convex optimization problem. We briefly discuss *direct loss minimization* in Section 5.7. For parameter estimation it is often advisable to include a regularization term $\Omega(\boldsymbol{w})$ which prevents overfitting. ERM in

empirical risk minimization

surrogate loss

3 Note that we use the sup-index notation to refer to an index rather than to the exponentiation.

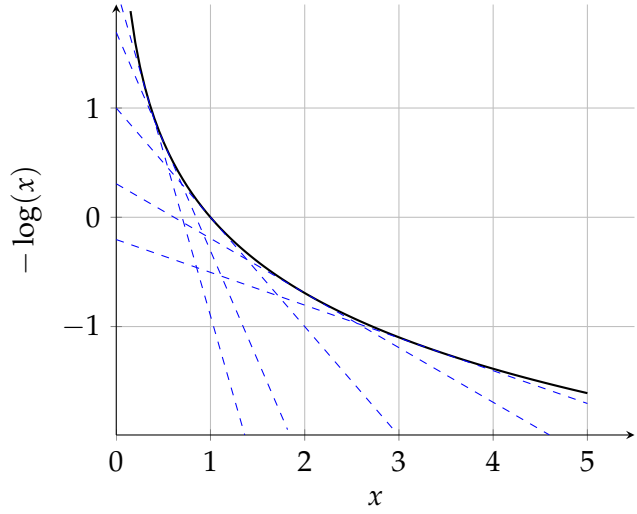


Figure 2.7: Idealized illustration of the Legendre-Fenchel transformation for the convex function $-\log(x)$. Through its conjugate dual, the negative logarithm can be rewritten as a maximization problem over linear functions (dashed lines).

these settings can be written as the following optimization problem for choosing the parameter $\boldsymbol{w}$:

$$\boldsymbol{w}^\star = \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{w}, \boldsymbol{x}^n, \boldsymbol{y}^n) + \lambda \Omega(\boldsymbol{w}). \tag{2.17}$$

regularization

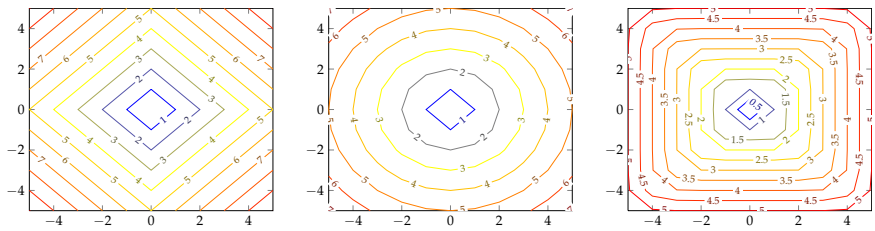
Here $\lambda \in \mathbb{R}^+$ denotes the weight of the regularizer. For the regularization we usually assume an L_1 or L_2^2 norm, i.e. $|\boldsymbol{w}|$ or $\frac{1}{2}\|\boldsymbol{w}\|_2^2$ as they both lead to convex optimization problems (assuming the surrogate loss is convex). More generally the L_p norm can be used[4]. For a vector $\boldsymbol{a} \in \mathbb{R}^D$ and $p \geq 1$ the p-norm is defined as

$$\|\boldsymbol{a}\|_p := \left(\sum_{d=1}^{D} |a_d|^p \right)^{1/p}.$$

Figure 2.8 visualizes the L_p-norm for different values of p. Next, we introduce the two most popular surrogate losses for structured outputs and their extensions to scenarios where a subset of the output variables

4 When employing the L_p-norm in a learning algorithm, often the exponentiated form L_p^p is considered as it is easier to compute.

is unobserved. A unifying theory linking these two approaches is presented in Chapter 5.



(a) $p = 1$ (b) $p = 2$ (c) $p = 10$

Figure 2.8: L_p for different values of p in the two-dimensional case. For large values of p the L_p-norm eventually becomes the maximum-norm, taking the value of the maximum element.

2.5.1 *Maximum Likelihood Learning*

Conditional Random Field

The Conditional Random Field (CRF) (Lafferty, McCallum, and Pereira 2001) initially proposed for natural language processing applications is a popular model for structured data. The conditional distribution of outputs given inputs is modeled using the Gibbs distribution as in (2.11). Learning then consists of maximizing the likelihood

$$\boldsymbol{w}^\star = \operatorname*{argmax}_{\boldsymbol{w}} \left(\prod_{n=1}^{N} P(\boldsymbol{y}^n | \boldsymbol{x}^n, \boldsymbol{w}) \right)^{1/N}. \tag{2.18}$$

This is equivalent to minimizing the negative log-likelihood and hence the surrogate loss in the ERM framework is given by

$$\ell_{ll}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) := -\log P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = -\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle + \log Z(\boldsymbol{x}, \boldsymbol{w}).$$

In general, ERM for the log-likelihood does not have a closed-form solution and hence in practice numerical solvers are used. These solvers rely on derivatives computations for efficiently solving the minimization problem. The derivative of the loss w.r.t. the parameters $\boldsymbol{w}$ is given by

$$\frac{\partial \ell_{ll}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{w}} = -\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) + \mathbb{E}_{P(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{w})}[\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}')]. \tag{2.19}$$

Here the second part of the gradient denotes the expectation of the feature map w.r.t. the Gibbs distribution. The expected feature map

follows from the property that for an exponential family distribution the derivatives of the partition function generate all the moments of the sufficient statistics, see Section 2.4. The expectation computation factorizes over the individual factors and requires the marginals thereof:

$$\mathbb{E}_{P(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{w})}[\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}')] = \sum_{t\in\mathcal{T}}\sum_{c\in\mathcal{C}(t)}\sum_{\boldsymbol{y}'_c} P(\boldsymbol{y}'_c|\boldsymbol{x},\boldsymbol{w})\boldsymbol{\phi}_t(\boldsymbol{x},\boldsymbol{y}'_c,c).$$

In most cases the second derivative is not used for the numerical optimization due to its size being quadratic in the dimension of the sufficient statistics. Thanks to the moment generating property of exponential families the second derivative is simply

$$\frac{\partial^2 \ell_{ll}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{w}^2} = \mathrm{Cov}_{P(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{w})}[\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}')].$$

As the covariance is always positive semidefinite it readily follows that ERM with the log-loss is a *convex minimization problem*. Maximum likelihood learning with the L_1- or the L_2^2-norm as a regularizer can also be understood as MAP parameter estimation with a Laplacian or a Gaussian prior distribution, respectively.

Alternatively, maximum likelihood estimation in CRFs also has an interpretation as an *entropy maximization* under the constraint that the expected sufficient statistics under the model $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w})$ match the observed statistics. For a dataset $\{(\boldsymbol{x}^n,\boldsymbol{y}^n)\}$, the (non-regularized) maximum likelihood objective can be written as

$$\min_{\boldsymbol{w}} \sum_{n=1}^{N} \ell_{ll}(\boldsymbol{w},\boldsymbol{x}^n,\boldsymbol{y}^n) = \min_{\boldsymbol{w}} -\sum_{n=1}^{N}\langle \boldsymbol{w},\boldsymbol{\phi}(\boldsymbol{x}^n,\boldsymbol{y}^n)\rangle + \sum_{n=1}^{N}\log Z(\boldsymbol{x}^n,\boldsymbol{w}).$$

By conjugate duality this can be rewritten as an entropy maximization under a constraint on the expected feature map:

$$\begin{aligned}\max_{\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^n\in\mathcal{M}\times\ldots\times\mathcal{M}} \quad & \sum_{n=1}^{N} H(P(\boldsymbol{y}|\bar{\boldsymbol{\theta}}(\boldsymbol{x}^n,\boldsymbol{\mu}^n))) \\ s.t. \quad & \sum_{n=1}^{N} P(\boldsymbol{y}|\bar{\boldsymbol{\theta}}(\boldsymbol{x}^n,\boldsymbol{\mu}^n))\boldsymbol{\phi}(\boldsymbol{x}^n,\boldsymbol{y}) = \sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{x}^n,\boldsymbol{y}^n).\end{aligned}$$

For the numerical minimization of ERM with the log-loss and an L_2 regularization, the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm (Liu and Nocedal 1989; Nocedal and Wright 1999), a

quasi-Newton method which approximates the Hessian, is most often used in practice. Historically, sometimes also the Conjugate Gradient method (Shewchuk 1994) has been used. If a L_1 regularization is desired, then the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) (Andrew and Gao 2007), a quasi-Newton solver specifically designed to handle the non-differentiable L_1 regularizer was shown to lead to good convergence properties and also several open-source implementations are available by now. Another popular solver, especially for large training data sets, is stochastic gradient descent (Robbins and Monro 1951; Kiefer and Wolfowitz 1952). It is applicable to L_1 as well as L_2 regularization.

For some models, either due to the measurement process or because of modeling assumptions it can happen that some of the output variables are unobserved. The Hidden Conditional Random Field (HCRF) (Quattoni et al. 2007) is an extension of the CRF to models where in training some of the output variables are hidden or unobserved. Unobserved variables are sometimes also referred to as latent variables. We will denote the hidden variables by $\boldsymbol{z} \in \mathcal{Z}$. Again we follow the convention of using a subscript $z_i \in \mathcal{Z}_i$ to index the i-th hidden variable. The HCRF models the joint conditional distribution of the hidden variables $\boldsymbol{z}$ and the output variables $\boldsymbol{y}$ given the input variables:

hidden variables

$$P(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{w})} \exp(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle).$$

The partition function now also sums over the states of $\boldsymbol{z}$:

$$Z(\boldsymbol{x}, \boldsymbol{w}) = \sum_{\boldsymbol{y} \in \mathcal{Y}, \boldsymbol{z} \in \mathcal{Z}} \exp(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle).$$

Moreover, the feature map is extended to also incorporate the hidden variables $\boldsymbol{z}$. For computing the posterior distribution of an output variables configuration, the hidden variables need to be marginalized out, to obtain

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{w})} \sum_{\boldsymbol{z} \in \mathcal{Z}} \exp(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle).$$

As for the CRF, learning then consists of maximizing the likelihood of the output variables given the input variables. Taking the negative logarithm to transfer the problem into an easier minimization problem,

we obtain the log-likelihood loss extended to partially unobserved domains

$$\ell_{ll}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = -\log \sum_{\boldsymbol{z} \in \mathcal{Z}} \exp(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle) + \log Z(\boldsymbol{x}, \boldsymbol{w}). \tag{2.20}$$

The first term can be understood as a restricted partition function where only hidden variables are summed over, whereas the second term sums over both output variables and hidden variables. The loss in (2.20) is no longer convex, as the first term is concave, but not convex. Therefore, in the general case, even if we could compute the partition functions, one can only hope to find a local minimizer.

2.5.2 *Maximum Margin Learning*

structured Support Vector Machine

The structured SVM (Tsochantaridis et al. 2005; Taskar, Guestrin, and Koller 2003) , sometimes also called maximum margin Markov network, can be understood as a non-probabilistic analogue to the CRF. Instead of estimating a distribution $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$ of outputs for a given input, the predictor itself is learned. The structured SVM considers a prediction function of the form

$$f(\boldsymbol{x}) := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle. \tag{2.21}$$

Prediction can be seen as a MAP estimate, where the parameters $\boldsymbol{w}$ might however not represent a meaningful posterior distribution. Learning of $\boldsymbol{w}$ directly trains with the prediction rule in (2.21) in mind. As in the standard binary SVM the classifier is trained such that it maximizes the margin between the ground-truth and any other output. The structured SVM is usually written as a Quadratic Program (QP):

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\xi}} \quad & \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 + \frac{1}{N} \sum_{n=1}^{N} \xi^n \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}^n) \rangle - \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}) \rangle \geq \Delta_{\boldsymbol{y}^n}(\boldsymbol{y}) - \xi^n \quad \forall \boldsymbol{y}, \forall n. \end{aligned} \tag{2.22}$$

Here ξ^n is a *slack variable* . The constraint says that we want all the outputs $\boldsymbol{y}$ to have at least a distance of $\Delta_{\boldsymbol{y}^n}(\boldsymbol{y})$ from the ground-truth $\boldsymbol{y}^n$[5]. An equivalent formulation to (2.22) is obtained when replacing

5 Note, that by definition $\Delta_{\boldsymbol{y}^n}(\boldsymbol{y}^n) = 0$ and therefore the ground-truth case does not need to be handled specially.

the margin constraint for all outputs by the constraint that has the maximum violation:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{N}\sum_{n=1}^{N}\xi^n \\ \text{s.t.} \quad & -\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}^n)\rangle + \max_{\boldsymbol{y}} \left[\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y})\rangle + \Delta_{\boldsymbol{y}^n}(\boldsymbol{y})\right] \leq \xi^n \quad \forall n. \end{aligned} \tag{2.23}$$

This formulation immediately gives rise to the *cutting-plane algorithm*, an alternating procedure in which the QP is solved for a current set of constraints. The constraint set is then augmented with the most violated constraint for the current estimate of $\boldsymbol{w}$. This algorithm is listed in Algorithm 2.1. Line 5 in Algorithm 2.1 corresponds to the

cutting-plane algorithm

Algorithm 2.1 Cutting-plane algorithm (Finley and Joachims 2008).

Require: $(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N), \lambda, \epsilon, \Delta_{\boldsymbol{y}^*}(\cdot)$.

1: $S^n \leftarrow \emptyset$ for $n = 1, \ldots, N$.
2: **repeat**
3: **for** $n = 1, \ldots, N$ **do**
4: $H(\boldsymbol{y}) := \Delta_{\boldsymbol{y}^n}(\boldsymbol{y}) + \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}) - \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}^n)\rangle$
5: compute $\hat{\boldsymbol{y}} = \operatorname{argmax}_{\boldsymbol{y}\in\mathcal{Y}} H(\boldsymbol{y})$
6: compute $\xi^n = \max\{0, \max_{\boldsymbol{y}\in S^n} H(\boldsymbol{y})\}$
7: **if** $H(\hat{\boldsymbol{y}}) > \xi^n + \epsilon$ **then**
8: $S^n \leftarrow S^n \cup \{\hat{\boldsymbol{y}}\}$
9: $\boldsymbol{w} \leftarrow$ optimize primal over $\bigcup_n S^n$
10: **end if**
11: **end for**
12: **until** no S^n has changed during iteration

loss augmented inference step . The output variable with the maximum score needs to be found. Note that, the linear score of the model is augmented with the loss term. Depending on the loss term this might render the original problem more difficult. Chapter 4 will discuss the loss augmented inference in more detail. The structured SVM in (2.22) rescales the margin by the loss term and is therefore called the *margin rescaled* structured SVM. There also exist the *slack rescaled* version of the structured SVM, which is covered in Chapter 5.

margin and slack rescaling

The structured SVM can be understood as employing the maximum margin loss in the ERM framework. The *maximum margin loss* is given by

$$\ell_{mm}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) := -\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle + \max_{\boldsymbol{y}' \in \mathcal{Y}} \left[\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}') \rangle + \Delta_{\boldsymbol{y}}(\boldsymbol{y}') \right]. \quad (2.24)$$

latent structured SVM

As for the CRF, there also exists an extension of the structured SVM to hidden variables scenarios. The corresponding classifier is then referred to as the *latent structured SVM* (Felzenszwalb, McAllester, and Ramanan 2008; Yu and Joachims 2009; Felzenszwalb et al. 2010). The prediction rule in (2.21) becomes

$$f(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \max_{\boldsymbol{z} \in \mathcal{Z}} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle.$$

Furthermore, the maximum margin loss for learning becomes

$$\ell_{mm}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = -\max_{\boldsymbol{z} \in \mathcal{Z}} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle + \max_{\substack{\boldsymbol{y}' \in \mathcal{Y} \\ \boldsymbol{z} \in \mathcal{Z}}} \left[\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{z}) \rangle + \Delta_{\boldsymbol{y}}(\boldsymbol{y}') \right]. \quad (2.25)$$

As for the HCRF, ERM with the maximum margin loss in a partially unobserved scenario leads to a non-convex optimization problem.

2.5.3 *Comparison of CRF and Structured SVM*

Figure 2.9 contrasts the two learning and prediction approaches, with CRF and Minimum Bayes Risk prediction on the one hand and structured SVM learning and MAP prediction on the other hand. A much more in-depth discussion is given in Chapter 5.

2.6 STRUCTURED MODELS

This section moves away from the abstract feature map $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ view point taken in the previous sections and gives concrete instantiations of $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ for many important applications of machine learning. We generally state the simplest possible model for a particular task which still illustrates the important concepts. Therefore we usually do not model any data dependence for the pairwise interactions, although this would be a straightforward extension. A reader who is already familiar with the joint input/output feature map $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ can safely skip this section, as it does not introduce any new notation.

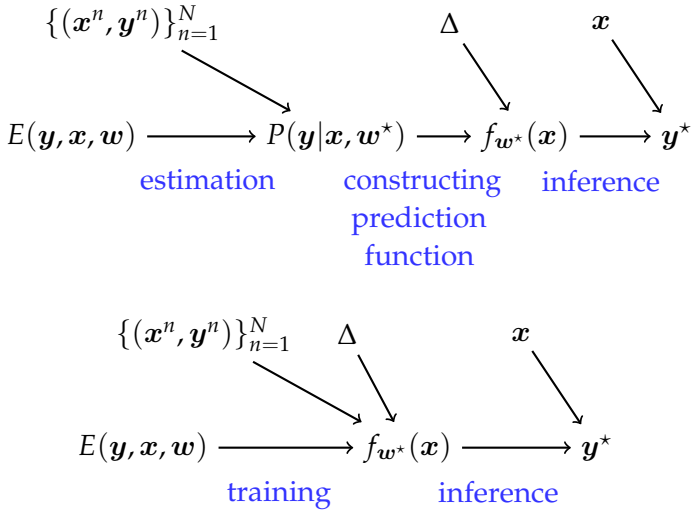


Figure 2.9: The two learning and prediction approaches. *Top*: The classical approach first estimates a posterior $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}^\star)$ from training data and incorporates the loss Δ at test-time to infer the optimal label. *Bottom*: The alternative approach directly trains a classifier for a specific loss and skips the distributional estimation step.

2.6.1 *Binary Classification*

Binary classification is probably the best studied problem in machine learning. As we will show, modern, yet simple algorithms such as the SVM (Boser, Guyon, and Vapnik 1992; Cortes and Vapnik 1995) or logistic regression can be understood in the framework introduced so far. For $y \in \{-1, 1\}$ the sufficient statistics can simply be chosen as

$$\boldsymbol{\phi}(\boldsymbol{x}, y) = \frac{1}{2} y \boldsymbol{\phi}(\boldsymbol{x}).$$

Here $\boldsymbol{\phi}(\boldsymbol{x})$ is the standard mapping known from the support vector machines literature from the original input spaces to a possibly much higher-dimensional space. Alternatively, it can be thought of as a sufficient statistics of the data. For the zero-one loss $\Delta_{y^\star}(y) = 1 - \mathbb{I}_{y^\star}(y)$, maximum-margin learning according to (2.24) with $\boldsymbol{\phi}(\boldsymbol{x}, y)$ is equivalent to the standard SVM formulation: Elementary calculations show

that the *Hinge loss* $\max(0, 1 - \langle \boldsymbol{w}, y\boldsymbol{\phi}(\boldsymbol{x}) \rangle)$ is obtained. On the other hand, if the logarithmic loss is used, the logistic regression approach is identified. This can be checked as follows; let us denote the inner product by $s_y = \langle \boldsymbol{w}, \frac{1}{2} y\boldsymbol{\phi}(\boldsymbol{x}) \rangle$, the posteriori is then given by

$$\begin{aligned} P(y|\boldsymbol{x}, \boldsymbol{w}) &= \frac{\exp(s_y)}{\exp(s_y) + \exp(-s_y)} \\ &= \frac{1}{1 + \exp(-\langle \boldsymbol{w}, y\boldsymbol{\phi}(\boldsymbol{x}) \rangle)}. \end{aligned}$$

This is equivalent to the standard definition of logistic regression. The approach above does not have a bias term b, usually present in the SVM formulation. A bias term can be included by adding a constant, say 1, to the sufficient statistics $\boldsymbol{\phi}(\boldsymbol{x})$. The corresponding element in the parameter vector $\boldsymbol{w}$ can then be thought of as the bias term. This is a standard approach, sometimes referred to as representing the data in a *homogeneous coordinate system*. There is one technical difficulty with this approach though, as the parameter also incurs regularization costs, which is generally not desirable for the bias term and not done for b in the standard binary SVM.

2.6.2 *Multiclass Classification*

As for the binary classification setting, let us assume we are given a data point $\boldsymbol{x}$ and a corresponding sufficient statistics $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{\bar{D}}$. Further, the output variable y is a categorical variable taking values in $\{1, \ldots, K\}$, which one aims to predict. The joint sufficient statistics between input and output variables is then

$$\boldsymbol{\phi}(\boldsymbol{x}, y) = \begin{bmatrix} \mathbb{I}_2(y)\boldsymbol{\phi}(\boldsymbol{x}) \\ \vdots \\ \mathbb{I}_K(y)\boldsymbol{\phi}(\boldsymbol{x}) \end{bmatrix}. \tag{2.26}$$

The joint sufficient statistics $\boldsymbol{\phi}(y, \boldsymbol{x})$ therefore has dimensionality $\bar{D} \cdot (K - 1)$, where $\bar{D}$ is the dimensionality of the data sufficient statistics $\boldsymbol{\phi}(\boldsymbol{x})$. For the feature map in (2.26), the log-likelihood and max-margin learning approach lead to the multiclass extensions of logistic regression and the SVM (Crammer and Singer 2002). For the SVM there exist several multiclass extensions, such as one-versus-all or one-versus-one, but the extension by Crammer and Singer (2002) is generally considered the

most sound one. In (Crammer and Singer 2002), the standard zero-one loss is used for $\Delta_{y^\star}(y)$. In case some similarity between the classes exists, one can include this information through the loss term. Note that the multiclass SVM in (Crammer and Singer 2002) has $\bar{D} \cdot K$ parameters instead of the $\bar{D} \cdot (K-1)$ parameters in the formulation here. It can be easily checked that one class can just be assumed to have an inner product of zero without changing the expressivity of the classifier. This is similar to the binary case where the parameter space is $\bar{D}$ and not $2\bar{D}$.

2.6.3 *Multilabel Classification*

A related problem to multiclass classification is multilabel classification (Tsoumakas and Katakis 2007). A data point $\boldsymbol{x}$ can assume *several classes at the same time*. Here we denote the number of classes by $|\mathcal{V}|$. Formally, the label vector $\boldsymbol{y} \in \{-1, 1\}^{|\mathcal{V}|}$ encodes whether each of the labels is absent or present. Hence, the number of possible outputs for an input is $2^{|\mathcal{V}|}$. A simple model that incorporates the pairwise label *co-occurrence* is discussed in (Finley and Joachims 2008). Figure 2.10 visualizes the model as a graphical model. The co-occurrence information is modeled by edges in a fully connected graph over $|\mathcal{V}|$ nodes. The multilabel classification model can be summarized by a sufficient

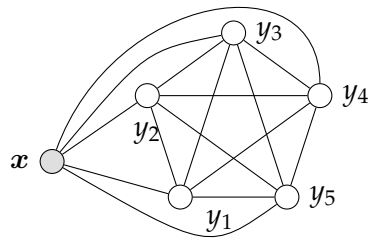


Figure 2.10: Pairwise graphical model for multi-label classification. Here for five possible labels.

statistics $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{\bar{D} \cdot |\mathcal{V}| + 3 \cdot |\mathcal{E}|}$ as follows:

$$\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in \mathcal{V}} \frac{1}{2} \boldsymbol{\phi}_i(\boldsymbol{x}) y_i + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\phi}_{ij}(y_i, y_j).$$

Here $\phi_i(\boldsymbol{x})$ is given by $\phi(\boldsymbol{x}) \in \mathbb{R}^{\bar{D}}$ copied to the i-th $\bar{D}$-dimensional block of $\phi(\boldsymbol{x}, \boldsymbol{y})$. $\phi_{ij}(y_i, y_j)$ simply encodes in which of the four possible states the edge (i, j) is. Each edge is associated with three entries[6] in the full sufficient statistics, which are set according to the state of the edge. For a one-dimensional $\phi(x) = x$ and two possible labels, the sufficient statistics for two example configurations are given as follows:

$$\phi(0.5, [-1, +1]) = \begin{bmatrix} -0.5 \\ 0.5 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \qquad \phi(0.1, [-1, -1]) = \begin{bmatrix} -0.1 \\ -0.1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Unlike in the binary and multiclass examples before, here the output variable $\boldsymbol{y}$ is multivariate and there exist dependencies among the individual y_i. As we will see, even though these dependencies lead to better predictions when compared with independent predictions, they render inference computations more difficult.

2.6.4 *Segmentation*

Segmentation is probably the most well-studied application of structured output prediction. In this thesis we will restrict ourselves to segmentation models for computer vision, but the models are equally important in other application domains, such as natural language processing. As an example we will study semantic segmentation, sometimes also called object recognition, which is considered an important intermediate step for high-level visual reasoning. The problem setup and task are illustrated in Figure 2.11. Many of the state-of-the-art methods (Ladicky et al. 2010; Rabinovich et al. 2007; Shotton et al. 2009) in computer vision are based on the same basic model, which is an extension of the discriminative Ising and Potts model discussed in Section 2.3. Also, the segmentation models share similarity to the multi-label classification model, with two notable differences. First, all

6 One out of the four states does not need to be explicitly considered in the sufficient statistics. This is similar to the multiclass model above: the energy can always be offset so that one of the states' contribution to the energy is zero.

Figure 2.11: Semantic segmentation on the Microsoft Research Cambridge (MSRC) dataset. *Left*: input image $\boldsymbol{x}$. *Middle*: segmentation $\boldsymbol{y}$ of the image. *Right*: Different colors encode different object categories.

the vertices and nodes are parameterized in the same way sharing the same parameters. Secondly, the graph is much sparser connected[7].

Formally let $\boldsymbol{y} \in \{1, \ldots, K\}^{|\mathcal{V}|}$ be the segmentation of an image, where $|\mathcal{V}|$ is equal to the number of pixels and K denotes the number of classes. The basic model consists of unary and pairwise terms, and the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given by the 4- or 8-connected grid, which was already discussed earlier in the context of the Ising model. The sufficient statistics is

$$\phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in \mathcal{V}} \phi_u(\boldsymbol{x}, y_i, i) + \sum_{(i,j) \in \mathcal{E}} \phi_p(y_i, y_j).$$

The unary sufficient statistics are similar to the multiclass case above, except that the features $\phi(\boldsymbol{x}, i)$ for the i-th pixel are extracted in a neighborhood around pixel i from the image $\boldsymbol{x}$.

$$\hat{\phi}_u(\boldsymbol{x}, y_i, i) = \begin{bmatrix} \mathbb{I}_2(y_i)\phi(\boldsymbol{x}, i) \\ \vdots \\ \mathbb{I}_K(y_i)\phi(\boldsymbol{x}, i) \end{bmatrix}.$$

For the features $\phi(\boldsymbol{x}, i)$, popular choices are color or gradient-based features such as the Histogram of Oriented Gradients (HOG) or the Scale-invariant feature transform (SIFT). Another choice is to use the class conditional probabilities from an already trained multiclass classifier, such as TextonBoost (Shotton et al. 2009), see e.g. (Ladicky 2011, Section

7 A recent exception is given in (Krähenbühl and Koltun 2011), where every pixel in an image is connected to all other pixels.

2.5.2) for more details. The pairwise sufficient statistics capture the $K^2 - 1$ co-occurrence statistics of class labels between neighboring pixels i and j

$$\hat{\phi}_p(y_i, y_j) = \begin{bmatrix} \mathbb{I}_{1,2}(y_i, y_j) \\ \mathbb{I}_{1,3}(y_i, y_j) \\ \vdots \\ \mathbb{I}_{2,1}(y_i, y_j) \\ \vdots \\ \mathbb{I}_{K,K}(y_i, y_j) \end{bmatrix}.$$

2.6.5 *Ranking*

The ranking problem is as follows: Given a set of $|\mathcal{V}|$ items, a user is asked to rank them according to his preferences. Ranking has widespread applications in information retrieval, where for example the results to a search query need to be returned to a user and sorted according to his preferences. Most often there is some knowledge about the user/query available, which is assumed to be encoded through the input variable $\boldsymbol{x}$. Moreover, some (possibly only partially labeled) training dataset of inputs and corresponding rankings is given. In the literature this problem is known as *learning to rank* (Liu 2009). We here discuss a structured model similar to (Xu et al. 2008). Let $y_i \in \{1, \ldots, |\mathcal{V}|\}$ denote the rank of the i-th item. The "ranking constraint", which says that $\boldsymbol{y}$ is a permutation of 1 to $|\mathcal{V}|$, can be expressed by a fully connected graphical model, with each edge (i, j) ensuring that $y_i \neq y_j$. The sufficient statistics has therefore the form

$$\phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in \mathcal{V}} \phi_i(\boldsymbol{x}, y_i) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}(y_i, y_j).$$

The unary sufficient statistics is the same as in the multiclass case and the pairwise sufficient statistics is defined by

$$\hat{\phi}_{ij}(y_i, y_j) = \begin{bmatrix} \mathbb{I}_{[y_i > y_j]}(y_i, y_j) \\ \mathbb{I}_{[y_i = y_j]}(y_i, y_j) \end{bmatrix}.$$

Here we extended the indicator function to more general Boolean expressions, rather than simply checking for equality. The parameters corresponding to the equality part should be set to $-\infty$, in order to ensure that the labeling is a valid ranking.

2.6.6 *Multiple-Instance Learning*

Multiple-Instance Learning has initially been proposed in the 1990s (Dietterich, Lathrop, and Lozano-Pérez 1997; Auer 1997; Long and Tan 1998). In particular, two large-margin approaches have been introduced in (Andrews, Tsochantaridis, and Hofmann 2002), which we shall focus on here. Variants of the multiple-instance SVMs have recently been rediscovered in the context of latent SVMs in computer vision (Felzenszwalb et al. 2010). The basic setting in multiple-instance learning is as follows: We are given a *bag* of data points $\{\boldsymbol{x}_i\}_{i=1}^I$ for which a label $y \in \{-1, 1\}$ is provided in training. The important difference to standard supervised learning is the fact that the label is given for the whole bag, but not for the individual instances in a bag. The semantic meaning of the bag-level label is: If $y = 1$, then at least one of the instances in the bag is positive, if $y = -1$ then all the instances are negative. In this sense the information provided by the label is asymmetric, as a negative label implicitly reveals the individual labels for all the instances in the bag, whereas a positive label only reveals the information that at least one instance has a positive label. Even though the models in (Andrews, Tsochantaridis, and Hofmann 2002) were initially not described in the structured output notation, we will show here that they can be understood as structured output models with hidden variables. (Andrews, Tsochantaridis, and Hofmann 2002) introduced two models, mi-SVM and MI-SVM, each with a slightly different point of view on the problem. Next, these multiple-instance models are given in our notation. Here $\phi_u(\boldsymbol{x}_i)$ will denote the sufficient statistics of the i-th instance.

bag

MICROSCOPIC MULTIPLE-INSTANCE SVM The mi-SVM[8] takes a microscopic approach and assigns to each individual instance in the bag a label. The bag label is positive if there is at least one positive instance label in the bag. As the instance level labels are never observed, these are modeled as hidden or latent variables $\boldsymbol{z}_i \in \{-1, 1\}^I$. The graphical model is given in Figure 2.12. The sufficient statistics is

mi-SVM

$$\phi(\boldsymbol{x}, y, \boldsymbol{z}) = \phi(y, \boldsymbol{z}) + \sum_{i=1}^{I} \phi_u(\boldsymbol{x}_i) z_i.$$

8 The form of the mi-SVM presented here is slightly different than the original one in the way that the margin is enforced. In the formulation here there is only one margin term for all instances in a bag, whereas the original formulation has one margin term for each instance.

$\phi(y, \boldsymbol{z})$ takes care of the multiple-instance constraint. If one of the hidden labels is active, then also y is positive:

$$\hat{\phi}(y, \boldsymbol{z}) = \begin{bmatrix} \mathbb{I}_{[y=1 \wedge \sum_{i=1}^{I} \frac{z_i+1}{2} < 1]}(y, \boldsymbol{z}) \\ \mathbb{I}_{[y=0 \wedge \sum_{i=1}^{I} \frac{z_i+1}{2} \geq 1]}(y, \boldsymbol{z}) \end{bmatrix}$$

Therefore the corresponding parameters should be $-\infty$ to exclude invalid configurations. If we wish to apply the structured SVM framework

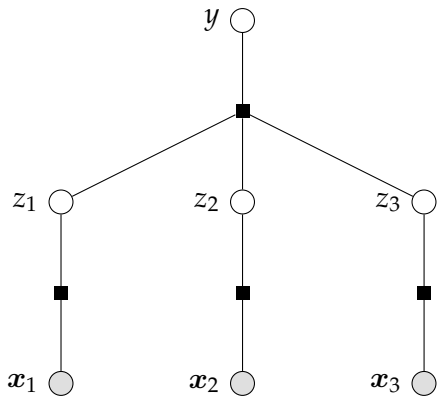


Figure 2.12: Graphical model for the mi-SVM.

to the mi-SVM model, we need to solve two problems, which turn out to be efficiently tractable:

1. Predict the hidden variables for a known bag label y

$$\boldsymbol{z} = \operatorname*{argmax}_{\boldsymbol{z}} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, y, \boldsymbol{z}) \rangle.$$

   For the case $y = -1$ this is easy, as this implies $\boldsymbol{z} = -\mathbf{1}$. For the case $y = 1$, one can predict the individual labels of the instances independently by $z_i = \operatorname{argmax}_{z_i} \langle \boldsymbol{w}, \boldsymbol{\phi}_u(\boldsymbol{x}_i, z_i) \rangle$. In case this leads to no positive instance label, the instance with the largest score is set to 1.

2. Prediction and loss-augmented inference

$$y, \boldsymbol{z} = \operatorname*{argmax}_{y, \boldsymbol{z}} [\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, y, \boldsymbol{z}) \rangle + \Delta_{y^n}(y)].$$

   Standard prediction is obtained as a special case by setting the loss term to zero. One can again predict the instance labels independently, and in case all labels are negative, the most positive

instance is set to 1. The score of the obtained positive solution is then compared to the solution where all labels are negative. Finally, the solution with the larger score is returned.

MACROSCOPIC MULTIPLE-INSTANCE SVM The MI-SVM takes a macroscopic approach and identifies a *witness*, an instance that is "the most positive", and uses the large-margin approach for this single instance. The concept of the witness is formalized through a single hidden variable $z \in \{1, \ldots, I\}$, which denotes the index of the witness instance. The sufficient statistics is given by (2.27) and the graphical model is shown in Figure 2.13.

MI-SVM

$$\phi(\boldsymbol{x}, y, z) = y \sum_{i=1}^{I} \phi(\boldsymbol{x}_i) \mathbb{I}_i(z) \tag{2.27}$$

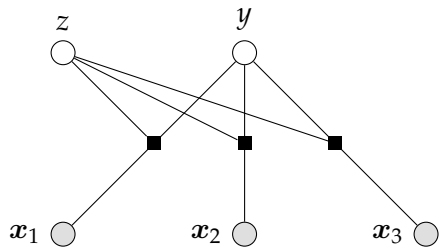


Figure 2.13: Graphical model for the MI-SVM.

Extensions of the binary multiple-instance models given here to settings where the label of a bag is multiclass or a multilabel are relatively straightforward, as the same ideas as in Subsection 2.6.2 and Subsection 2.6.2 can be used.

2.7 EXACT AND APPROXIMATE INFERENCE

So far we assumed black-box procedures for computing the MAP configuration or the partition function. In the current section we introduce exact and approximate algorithms for performing these tasks. We will drop the dependence of the Gibbs distribution on the parameters $\boldsymbol{w}$ as well as on the input variable $\boldsymbol{x}$. $\boldsymbol{w}$ and $\boldsymbol{x}$ are assumed to be fixed and therefore it is more convenient to think about a setting where the potentials are given explicitly rather then implicitly through $\boldsymbol{w}$ and $\boldsymbol{x}$.

Also, we restrict ourselves to a discussion of pairwise models, hence we study energies of the form (2.5):

$$E(\boldsymbol{y}) = \sum_{i \in \mathcal{V}} \theta_i(y_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(y_i, y_j).$$

Inference in graphical models can refer to different tasks, all of them generally suffering from the same problem: A maximization and/or integration over an exponentially large set has to be carried out. The most common questions that are summarized under the umbrella of inference in graphical models are:

Maximum-A-Posteriori

- Finding the Maximum-A-Posteriori (MAP) label. For Gibbs distributions of the form as in (2.11), which we study in this thesis, the computation reduces to an *energy minimization*:

$$\boldsymbol{y}^\star = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} P(\boldsymbol{y}) = \operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{Y}} E(\boldsymbol{y}).$$

  The term MAP implies that the distribution should be thought of as a valid posterior. The concept of energy minimization is more general and also valid if $E(\boldsymbol{y})$ does not necessarily represent a meaningful posterior. Therefore, nowadays the energy minimization problem is often referred to as finding the *Most Probable Explanation (MPE)* . We shall also adopt this convention here. The MPE problem is ubiquitous in graphical models. It for example arises when computing the prediction for a CRF or when solving the loss-augmented inference problem in the structured SVM.

most probable explanation

- The computation of the partition function $Z = \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp(\bar{E}(\boldsymbol{y}))$ is necessary when the likelihood is evaluated. The likelihood is for example required in the training of a CRF.

- The computation of marginals is another important inference problem. Let $\mathcal{S}$ correspond to a subset of the variables $\mathcal{V}$, then the marginal $P(\boldsymbol{y}_\mathcal{S})$ of this subset corresponds to the computation:

$$P(\boldsymbol{y}_\mathcal{S}) = \sum_{\boldsymbol{y}_{\mathcal{V} \setminus \mathcal{S}} \in \mathcal{Y}_{\mathcal{V} \setminus \mathcal{S}}} P(\boldsymbol{y})$$

  Here $\mathcal{V} \setminus \mathcal{S}$ corresponds to the output variables in the complement of $\mathcal{S}$ and $\boldsymbol{y}_{\mathcal{V} \setminus \mathcal{S}}$ the assignment to these variables. Finally, $\boldsymbol{y}$ is understood as the combination of $\boldsymbol{y}_{\mathcal{V} \setminus \mathcal{S}}$ and $\boldsymbol{y}_\mathcal{S}$. Of special interest

is the marginal of an individual variable i as well as marginals of variables sharing a common factor. As discussed in Section 2.1, the unary marginals $P(y_i)$ are important for computing the Maximum Posteriori Marginal (MPM) and the Minimum Mean Squared Error (MMSE) prediction in probabilistic models.

- Marginal MPE or marginal MAP (Koller and Friedman 2009, Section 2.1.5.3) refers to an inference problem where for some variables a maximization is performed and for other variables an integration is carried out. More formally, for a partition $(\mathcal{A}, \mathcal{B})$ of the variables $\mathcal{V}$, the marginal MPE corresponds to the task

$$\boldsymbol{y}_{\mathcal{A}}^{\star} = \operatorname*{argmax}_{\boldsymbol{y}_{\mathcal{A}}} \sum_{\boldsymbol{y}_{\mathcal{B}}} \exp(\bar{E}(\boldsymbol{y})).$$

  The marginal MPE problem arises for example in the context of minimum Bayes risk prediction as discussed in Section 2.1 as the unknown true state of the output variables is marginalized over, see (2.1). We will also be confronted with the marginal MPE problem in Chapter 6.

2.7.1 *Hardness of Inference*

Deciding whether a labeling is the MPE configuration is NP-hard. For the related problem of inference in Bayesian networks, this fact is for example shown in (Cooper 1990; Roth 1996). Chandrasekaran, Srebro, and Harsha (2008) discuss several inference problems and their complexity.

The decision variant of MPE inference can be shown to be NP-hard by reducing 3-SAT to MPE inference. The reduction introduces a variable for each literal and each clause maps to one factor. The potential is 0 if the clause is fulfilled and ∞ otherwise. Deciding whether a formula has a satisfying assignment is then equivalent to deciding whether a MPE solution with zero energy exists.

The computation of the partition function of an undirected graphical model on the other hand is a #P-hard problem. This fact follows from a reduction of the #P-hard problem of computing the permanent (Valiant 1979) to a partition function computation in an undirected graphical model.

2.7.2 *Inference for a Tree and Belief Propagation*

Trees have the favorable property that no loops are present in the graph. It turns out that this property renders many of the inference tasks tractable. We will restrict the discussion here to pairwise graphical models and tree graphs. The algorithms can also be extended to general loop-free factor graphs or graphical models with a forest structure. One of the main difficulties of probabilistic inference is posed by the fact that the joint distribution $P(\boldsymbol{y})$ often does not permit a representation by *marginals* $P(y_i)\ \forall i$ and $P(y_i, y_j)\ \forall (i,j) \in \mathcal{E}$. In the remainder of this thesis we use $\mu_i(y_i) := P(y_i)$ and $\mu_{ij}(y_i, y_j) := P(y_i, y_j)$. The joint distribution for a tree-shaped graphical model builds the notable exception and can be written as

$$\begin{aligned} P(\boldsymbol{y}) &= \prod_{i \in \mathcal{V}} \mu_i(y_i) \prod_{(i,j) \in \mathcal{E}} \frac{\mu_{ij}(y_i, y_j)}{\mu_i(y_i)\mu_j(y_j)} \\ &= \prod_{i \in \mathcal{V}} \mu_i(y_i)^{1-d_i} \prod_{(i,j) \in \mathcal{E}} \mu_{ij}(y_i, y_j). \end{aligned} \tag{2.28}$$

Here d_i denotes the degree of the i-th node in the graphical model. And we define $0/0 := 0$. (2.28) is a consequence of the so-called *junction-tree theorem* (Wainwright and Jordan 2008, Proposition 2.1). Due to this factorization, using simple algebra and exchanging the order of the sums, the entropy of the joint distribution can be shown to be given by

junction-tree theorem

$$\begin{aligned} H(P) &:= -\sum_{\boldsymbol{y} \in \mathcal{Y}} P(\boldsymbol{y}) \log P(\boldsymbol{y}) \\ &= \sum_{i \in \mathcal{V}} (1 - d_i) H(\boldsymbol{\mu}_i) + \sum_{(i,j) \in \mathcal{E}} H(\boldsymbol{\mu}_{ij}). \end{aligned} \tag{2.29}$$

Here $H(\boldsymbol{\mu}_i)$ and $H(\boldsymbol{\mu}_{ij})$ denote the unary and pairwise entropies. We assume that the unary and pairwise marginals are represented as a vector of dimensionality equal to $|\mathcal{Y}_i|$ and $|\mathcal{Y}_i| \cdot |\mathcal{Y}_j|$, respectively. For a probability vector $\boldsymbol{\mu}$ over K states, the entropy is defined as

$$H(\boldsymbol{\mu}) := -\sum_{k=1}^{K} \mu_k \log \mu_k.$$

For the unary marginals, k runs over all the states in $\mathcal{Y}_i$, whereas for the pairwise marginals, the index runs over all joint configurations $\mathcal{Y}_i \times \mathcal{Y}_j$. The tree entropy in (2.29) can alternatively also be written as

$$H(P) = \sum_{i \in \mathcal{V}} H(\boldsymbol{\mu}_i) + \sum_{(i,j) \in \mathcal{E}} I_{ij}(\boldsymbol{\mu}_{ij}, \text{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\top)),$$

where $\text{vec}(\cdot)$ denotes the "vectorization" of a matrix and I_{ij} denotes the *mutual information* for edge (i, j):

$$I_{ij}(\boldsymbol{\mu}_{ij}, \text{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\top)) = \sum_{(y_i, y_j) \in \mathcal{Y}_i \times \mathcal{Y}_j} \mu_{ij}(y_i, y_j) \log \left(\frac{\mu_{ij}(y_i, y_j)}{\mu_i(y_i) \mu_j(y_j)} \right). \quad (2.30)$$

The mutual information is non-negative and only equal to zero if the factorized and joint distribution are the same. The mutual information can be expressed as

$$I_{ij}(\boldsymbol{\mu}_{ij}, \text{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\top)) = H(\boldsymbol{\mu}_i) + H(\boldsymbol{\mu}_j) - H(\boldsymbol{\mu}_{ij}). \quad (2.31)$$

The factorization properties for tree-shaped graphical give rise to exact dynamic programming approaches for the inference of the marginals and the MPE configuration. The algorithms are widely known in different fields under different names: belief-propagation (Pearl 1986), the Viterbi algorithm (Viterbi 1967) and the forward-backward algorithm (Rabiner 1989) are all based on the same idea. The algorithm is given in Algorithm 2.2 and has the same form for the inference of marginals and the MPE configuration, it only differs in the way the messages and the final solution are computed.

Algorithm 2.2 Message-passing for a tree-shaped graphical model.

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\theta_i \ \forall i \in \mathcal{V}$, $\theta_{ij} \ \forall (i,j) \in \mathcal{E}$.

1: initialize the messages $m_{i \to j}(y_j) \ \forall (i,j) \in \mathcal{E}, y_j \in \mathcal{Y}_j$.
2: define a node $r \in \mathcal{V}$ to be the root.
3: send messages $m_{i \to j}$ upwards from the leafs to the root r.
4: send messages $m_{i \to j}$ downwards from the root r to the leafs.
5: **return** marginals according to (2.33) and (2.34) or MPE assignment according to (2.35).

For inference of the marginals, the sum-product updates are used:

$$m_{i \to j}(y_j) \propto \sum_{y_i} \left[\exp(\bar{\theta}_{ij}(y_i, y_j) + \bar{\theta}_i(y_i)) \prod_{s \in \mathcal{N}(i) \setminus j} m_{s \to i}(y_i) \right].$$

sum-product algorithm

Algorithm 2.2 with the sum-product updates is commonly referred to as the *sum-product algorithm*. On the other hand, for inference of the MPE configuration, the max-product updates are employed instead:

$$m_{i\to j}(y_j) \propto \max_{y_i} \left[\exp(\bar{\theta}_{ij}(y_i, y_j) + \bar{\theta}_i(y_i)) \prod_{s\in\mathcal{N}(i)\setminus j} m_{s\to i}(y_i) \right]. \tag{2.32}$$

max-product algorithm

Algorithm 2.2 with the max-product updates is called the *max-product algorithm*. The solution is computed differently for the two problems. For the sum-product algorithm, the unary marginals are computed as follows:

$$\mu_i(y_i) \propto \exp(\bar{\theta}_i(y_i)) \prod_{j\in\mathcal{N}(i)} m_{j\to i}(y_i), \tag{2.33}$$

and the pairwise marginals as

$$\mu_{ij}(y_i, y_j) \propto \exp(\bar{\theta}_{ij}(y_i, y_j)) \frac{\mu_i(y_i)\mu_j(y_j)}{m_{i\to j}(y_j) m_{j\to i}(y_i)}. \tag{2.34}$$

On the other hand, the MPE assignment is obtained by

$$y_i = \operatorname*{argmax}_{y_i} \prod_{j\in\mathcal{N}(i)} m_{j\to i}(y_i). \tag{2.35}$$

This computation can be performed at the root as soon as all incoming messages are known. For max-product it is important to note that one needs to keep track of the maximizing variables in (2.32). Like this in the down-wards pass of the message-passing, the computation of the MPE assignment according to (2.35) can be carried out consistently in case there exist several maximizing configurations.

Both algorithms have a linear complexity in the number of nodes $|\mathcal{V}|$ and the number of edges $|\mathcal{E}|$ and a quadratic complexity in the number of states K^2, where K denotes the size of the variable with the largest output domain. Altogether the complexity is hence $\mathcal{O}(K^2|\mathcal{E}| + K|\mathcal{V}|)$. Note that the $\mathcal{O}$-notation does not absorb large constants, as only two passes through the tree are required. To appreciate these results, we would like to point out that a naive MPE inference algorithm would simply compute the energy for each of the exponentially many configurations $\boldsymbol{y} \in \mathcal{Y}$ and return the one with the lowest energy. This naive algorithm would result in a complexity of the form $\mathcal{O}(K^{|\mathcal{V}|} \cdot (|\mathcal{V}| + |\mathcal{E}|))$. A slightly modified version of this naive algorithm could also be

used for the marginals computation, resulting in the same exponential complexity.

Algorithm 2.2 gives an efficient and exact way to compute the marginals or the MPE configuration of a tree-shaped graphical model. However, what about the partition function? As the partition function can be related to the entropy and the internal energy according to (2.4), the sum-product algorithm also immediately gives a tractable approach to compute the partition function: One first computes the marginals of the nodes and edges in the graphical model, from this it is then easy to compute the internal energy and the entropy. Finally from (2.4) the partition function can then be computed.

Algorithm 2.2 will only work for trees, as in a graph with loops the root is not well-defined. A modified algorithm, called *loopy belief propagation* (Pearl 1990; Yedidia, Freeman, and Weiss 2005) computes the same messages, but does not perform a topological sort of the nodes first. Loopy belief propagation exists in various variants, such as synchronous and asynchronous schedules (Elidan, McGraw, and Koller 2006). Loopy belief propagation is still converging to the marginals or the MPE assignment for trees, but is no longer guaranteed to only require two iterations. In many applications, loopy belief propagation has been found to lead to satisfying approximate solutions, and is still popular due to its simplicity.

loopy belief propagation

The discussion of message-passing is on purpose kept short. For more detailed explanations and exact computations in factor graphs, see (Bishop 2006, Chapter 8) or (MacKay 2003, Chapter 26). (Hazan and Shashua 2010) gives a unifying view on many different variants of message-passing.

2.7.3 *Variational Inference*

Section 2.4 introduced the variational formulation of the log partition function. This variational formulation can be used for both, the computation of the partition function (and hence the marginals), but also for the evaluation of the MPE assignment. By conjugate duality, both problems can be stated as the following minimization problem:

$$\min_{\boldsymbol{\mu} \in \mathcal{M}} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle - \frac{1}{\beta} H(\boldsymbol{\mu}). \tag{2.36}$$

For MPE inference, $\beta \to \infty$ and hence the entropy can be discarded from the objective. As stated before, there exist two problems with the formulation above:

1. The marginal polytope $\mathcal{M}$ is difficult to describe and requires an exponential number of constraints in general.

2. For the computation of marginals, when the entropy term does not vanish there is an additional obstacle: The entropy does in general not factorize over the marginals and is therefore difficult to compute[9]. The tree case is a notable exception, where due to the junction tree theorem it is possible to express the joint distribution through the marginals, see (2.28).

upper and lower bounds

A large body of research has considered approximations of the variational problem above, by tackling these two obstacles. We differentiate two classes of approximation approaches based on whether for the MPE problem the obtained solutions are upper or lower bounds on the true energy of the MPE assignment. The first class of algorithms relaxes the marginal polytope $\mathcal{M}$ and therefore obtains lower bounds on the true MPE configuration. On the other hand, a second class of algorithms leads to upper bounds on the MPE configuration. We refer to the former class as an outer approximation, to the latter as an inner approximation. We briefly sketch the ideas below, and review the approaches in more detail in Chapter 3.

2.7.4 *Outer Approximations*

local marginal polytope

Outer Approximations replace the combinatorially large marginal polytope $\mathcal{M}$ by the simpler *local marginal polytope* $\mathcal{L}_\mathcal{G}$. In general $\mathcal{L}_\mathcal{G}$ also contains marginal configurations that do not correspond to any valid joint distribution $P(\boldsymbol{y}|\bar{\boldsymbol{\theta}})$. $\mathcal{L}_\mathcal{G}$ only includes local summation and normalization constraints and is formally specified by

$$\mathcal{L}_\mathcal{G} = \left\{ \boldsymbol{\mu} \;\middle|\; \begin{array}{ll} \sum_{y_i} \mu_i(y_i) = 1 & \forall i \in \mathcal{V} \\ \sum_{y_j} \mu_{ij}(y_i, y_j) = \mu_i(y_i) & \forall y_i, (i,j) \in \mathcal{E} \\ \sum_{y_i} \mu_{ij}(y_i, y_j) = \mu_j(y_j) & \forall y_j, (i,j) \in \mathcal{E} \\ \mu_{ij}(y_i, y_j) \geq 0 & \forall y_i, y_j, (i,j) \in \mathcal{E} \end{array} \right\}. \tag{2.37}$$

9 Again, note that $H(\boldsymbol{\mu})$ should be read as "the entropy of the distribution that gave rise to the marginals $\boldsymbol{\mu}$".

For the special case, where there exist no loops in the graphical model, it holds that $\mathcal{M} = \mathcal{L}_\mathcal{G}$. The equivalence follows from the fact that for a tree-shaped graph, the junction-tree property ensures the expressibility of the joint distribution as a product of pairwise and unary marginals.

One immediately obtains an approach to compute a lower bound on the energy of the MPE solution by replacing the marginal polytope by the local marginal polytope and solving the resulting linear program. In the case of marginal inference, i.e., when the entropy in (2.36) does not vanish, one additionally needs to upper bound the entropy.

2.7.5 *Inner Approximations and Meanfield Methods*

The mean-field approach is originating from statistical physics; for a detailed review see (Wainwright and Jordan 2008, Section 5). Instead of relaxing the marginal polytope as in the outer approximations, for mean-field methods one restricts the possible marginals to a subset of the marginal polytope. The most popular approach is to restrict the family of marginals, to those that come from a *fully factorized distribution*, the product distribution. Contrary to the outer approximation discussed earlier, the mean-field approach generally leads to a computationally difficult non-convex optimization problem.

2.7.6 *Submodular Energies*

We have already seen that for tree-shaped graphical models inference is tractable and both marginals and the MPE configuration can be obtained in linear complexity in the number of edges and nodes. Another important family of tractable energies are *submodular functions* which are discrete analogues of convex functions (Fujishige 1991; Lovasz 1983). For submodular functions only the MPE computation is known to be tractable, no similar results are known for marginals. Submodular functions are particularly important because of their widespread use in modeling labeling problems in computer vision such as 3D voxel segmentation (Snow, Viola, and Zabih 2000) and foreground-background image segmentation problems (Boykov 2001; Blake et al. 2004).

We refer to (McCormick 2006) for a more detailed review of submodular functions and their minimization. Let us consider a set $\mathcal{V}$ and

denote the size of this set by $n = |\mathcal{V}|$. A set function $f : 2^{\mathcal{V}} \to \mathbb{R}$ is called submodular if

$$f(\mathcal{S} + \{e\}) - f(\mathcal{S}) \geq f(\mathcal{T} + \{e\}) - f(\mathcal{T}) \quad \forall \mathcal{S} \subset \mathcal{T} \subset \mathcal{T} + \{e\}.$$

Here $\mathcal{S}$ and $\mathcal{T}$ denote subsets of $\mathcal{V}$. This property is often called diminishing return: Adding e to a larger set should increase the function value by no more than adding it to a smaller set. An equivalent definition of submodular functions is given by

$$f(\mathcal{X}) + f(\mathcal{Y}) \geq f(\mathcal{X} \cup \mathcal{Y}) + f(\mathcal{X} \cap \mathcal{Y}) \quad \forall \mathcal{X}, \mathcal{Y} \subseteq \mathcal{V}.$$

Minimization of general submodular functions is known to be polynomially time solvable. The currently fastest algorithm (Orlin 2009) has running time $\mathcal{O}(n^5 T + n^6)$, where T is the time required to evaluate the function and n is the number of elements in the set. These exact algorithms for submodular function minimization are hardly practical for the dimensions encountered in typical machine learning or computer vision problems. Therefore recent research in these fields has focused on two particular aspects of submodular function minimization. The first line of research identifies special submodular functions, for which efficient and exact algorithms can be devised (Kohli and Kumar 2010; Stobbe and Krause 2010), we will consider such a special case in Chapter 4. Another direction is to find general approximate submodular function minimization algorithms (Jegelka, Lin, and Bilmes 2011).

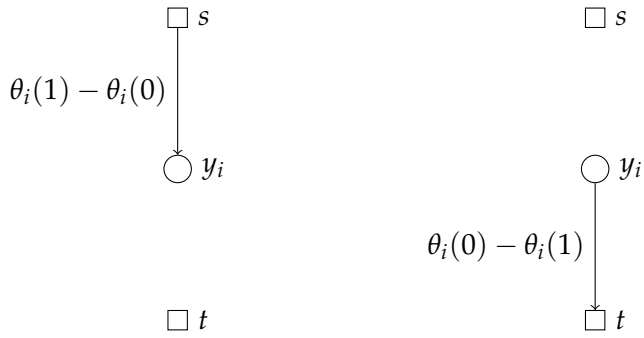
In case the function is pairwise and binary, the submodularity definition can be restated as follows. In terms of the energy formulation from earlier sections, the submodularity requirement reduces to

$$\theta_{ij}(0,0) + \theta_{ij}(1,1) \leq \theta_{ij}(0,1) + \theta_{ij}(1,0) \quad \forall (i,j).$$

In words this means that all the pairwise potentials have to be associative. There is no restriction on the unary potentials. It turns out that for the special case of submodular pairwise, binary functions the minimization can be restated as a minimum s-t-cut or equivalently as a maximum flow problem (Papadimitriou and Steiglitz 1982, Section 6.1). Given a *directed* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a cost c_{ij} for each edge $(i,j) \in \mathcal{E}$. Moreover, a dedicated source node s and a sink node t are specified. The minimum s-t-cut problem is to find a partition $(\mathcal{S}, \mathcal{T})$ of the nodes with $s \in \mathcal{S}$ and $t \in \mathcal{T}$, such that the cut-set cost $\sum_{(i,j)\in\mathcal{E}, i\in\mathcal{S}, j\in\mathcal{T}} c_{ij}$ is minimized. Below we discuss the construction of a

graph for which the solution of the minimum s-t-cut problem recovers the exact MPE assignment of a pairwise, binary, associative energy. In the computer vision literature it has been a common practice to refer to this construction as *graph-cut*. The graph-cut construction adopted here is from (Kolmogorov and Zabih 2004). Figure 2.14 visualizes the graph construction for a unary potential. This construction is carried out for

graph-cut



(a) Case 1, $\theta_i(0) < \theta_i(1)$. (b) Case 2, $\theta_i(0) \geq \theta_i(1)$.

Figure 2.14: Graph construction for the i-th node. Depending on which state has a lower energy, the node is connected to either the source s or the sink t.

every node $i \in \mathcal{V}$.

For each edge (i, j) a similar construction is carried out. Let us first denote the different terms of the edge energy as follows

$$\theta_{ij} = \begin{array}{|c|c|}\hline \theta_{ij}(0,0) & \theta_{ij}(0,1) \\ \hline \theta_{ij}(1,0) & \theta_{ij}(1,1) \\ \hline \end{array} = \begin{array}{|c|c|}\hline A & B \\ \hline C & D \\ \hline \end{array}$$

The pairwise energy can then be rewritten as

$$\begin{array}{|c|c|}\hline A & B \\ \hline C & D \\ \hline \end{array} = A + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline C-A & C-A \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & D-C \\ \hline 0 & D-C \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & B+C-A-D \\ \hline 0 & 0 \\ \hline \end{array}$$

The first term on the right hand side can be ignored as it is constant no matter what the assignment is. The second and third term can be represented by unary potentials for node i and j, respectively. The unary construction from above also needs to be applied to these potentials. Finally, the last term needs to be handled specially by a connection between the i-th and the j-th node. An example configuration is visualized in Figure 2.15. Note, that for associative energies $B + C - A - D$

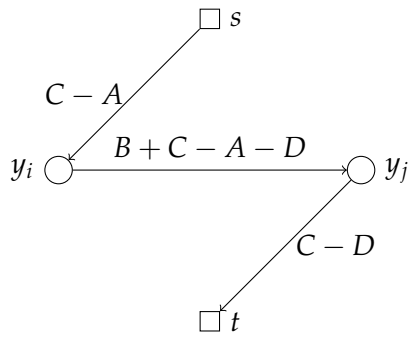


Figure 2.15: Pairwise graph-cut construction where for example $C > D$ and $C > A$.

is always positive and therefore none of the edges have a negative cost. The non-negativity is important as otherwise, maximum flow algorithms could not be applied, as non-negativity of flow is a crucial property.

Finally, as an alternative to the maximum flow algorithms, it can be shown that for pairwise, binary submodular energy functions, the Linear Program (LP) relaxation (for the local marginal polytope) leads to non-fractional solutions, and therefore to the global minima.

2.7.7 *Markov Chain Monte Carlo Techniques*

Gibbs sampling

Markov Chain Monte Carlo (MCMC) methods (Robert and Casella 2005) are widely adopted in practice and are based on *sampling*. We here restrict ourselves to *Gibbs sampling*, one of the simplest and most common approaches. Gibbs sampling in graphical models is based on the fact, that given the state of the Markov blanket of the i-th variable, it is easy to compute the probabilities for the different outcomes of y_i, as the partition function collapses to a sum over the $|\mathcal{Y}_i|$-many states, which is efficient to compute. We illustrate Gibbs sampling for a graphical model in Algorithm 2.3. An obvious application of Gibbs sampling is drawing typical configurations from a graphical model. However, variants can also be used for approximating marginals or computing an MPE assignment. For marginal inference, one can simply track the configurations drawn in several sweeps[10] and average to get an approx-

10 A sweep corresponds to sampling all variables once.

Algorithm 2.3 Gibbs sampling for a graphical model.

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\theta_i \; \forall i \in \mathcal{V}$, $\theta_{ij} \; \forall (i,j) \in \mathcal{E}$.
1: Initialize the state y_i randomly $\forall i \in \mathcal{V}$.
2: **while** not converged **do**
3: **for** $i = 1, \ldots, |\mathcal{V}|$ **do**
4: Sample y_i according to $P(y_i | \boldsymbol{y}_{\mathcal{N}(i)})$.
5: **end for**
6: **end while**

imation of the marginals. For MPE inference popular approaches are simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983) and deterministic annealing (Hofmann and Buhmann 1997). Annealing strategies increase the inverse temperature of the Gibbs distribution while performing Gibbs sampling, to eventually only sample from low-energy configurations.

2.8 APPROXIMATE LEARNING

As we have seen, learning with the maximum margin loss and the log-loss requires efficient algorithms to compute the partition function or infer the loss augmented MPE label. In general these two problems are intractable and therefore also learning is intractable. This section overviews the different approaches in the literature for approximate learning. An illustrative example for the need of choosing the combination of the inference algorithm and the learning objective carefully is given in (Kulesza and Pereira 2008): A learning algorithm is shown to fail, despite the fact that the approximate inference algorithm has strong approximation guarantees.

2.8.1 *Pseudolikelihood and Composite Likelihood*

For probabilistic learning of CRFs, a relatively well-established approach is given by maximum pseudolikelihood and maximum composite likelihood. The basic idea is to leverage on the known ground-truth observations in training by clamping variables in the graphical model to their observed state. Maximum pseudolikelihood was initially pro-

posed in (Besag 1975) and considers the following (here unregularized) learning objective

$$\boldsymbol{w}^\star = \max_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \sum_{i \in \mathcal{V}} \log P(y_i^n | \boldsymbol{y}_{\setminus i}^n, \boldsymbol{x}^n, \boldsymbol{w}).$$

Where $\boldsymbol{y}_{\setminus i}^n$ denotes all the variables, except the i-th variable of the n-th example. Conditioning on all the remaining variables is equivalent to conditioning on the variables in the Markov blanket of variable i. Because of the conditioning, the state space relevant for the computation of the partition function collapses to $\mathcal{Y}_i$, which is generally very small and therefore remains tractable. A generalization of pseudolikelihood is given by the composite likelihood (Lindsay 1988), which studies larger decompositions of the nodes $\mathcal{V}$ of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Formally, let $\{(\mathcal{A}_1, \mathcal{B}_1), \ldots, (\mathcal{A}_M, \mathcal{B}_M)\}$ denote a collection of M partitions of $\mathcal{V}$ into two sets, i.e. $\mathcal{A} \cup \mathcal{B} = \mathcal{V}$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$. Then composite likelihood [11] considers instead of the maximum likelihood objective the following objective

$$\boldsymbol{w}^\star = \max_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} \log P(\boldsymbol{y}_{\mathcal{A}_m}^n | \boldsymbol{y}_{\mathcal{B}_m}^n, \boldsymbol{x}^n, \boldsymbol{w}).$$

For certain choices of $\mathcal{A}$ and $\mathcal{B}$, the required partition function computation can still be carried out exactly. This is for example the case if $\mathcal{A}$ is chosen such that all loops are blocked by nodes in $\mathcal{B}$.

Both, maximum pseudolikelihood and maximum composite likelihood are concave optimization problems. By negating the objective, as before for maximum likelihood learning, general convex minimization algorithms for smooth objectives, such as L-BFGS, can be used to perform the numerical optimization. We will study different aspects of pseudolikelihood and composite likelihood in more detail in Chapter 6. Pseudolikelihood is generally only studied in the context of probabilistic models. A notable recent exception is (Sontag et al. 2010), which formulates a maximum pseudolikelihood objective for the training of structured SVMs. (Dillon and Lebanon 2010) recently introduced a stochastic version of composite likelihood, where the decompositions are chosen stochastically. With a small probability computationally more demanding decompositions, such as modest-sized cyclic subgraphs, are incorporated.

11 We restrict ourselves to the *conditional* composite likelihood; the related marginal conditional likelihood is not discussed here.

A desirable theoretical aspect of pseudolikelihood and composite likelihood is the property that in the limit as $N \to \infty$ and under some regularity conditions, the parameter estimate recovered by pseudolikelihood converges to the true value of the parameter (Lindsay 1988). Therefore, maximum pseudolikelihood is a *consistent* estimator. One disadvantage of pseudolikelihood in practice is however, that the variance of the estimator is large, and often very large datasets are needed in order to recover good parameter estimates.

consistency

2.8.2 *Contrastive Divergence*

Contrastive Divergence (CD) was initially introduced in (Hinton 2002) for the training of product of experts classifiers. CD applied to the training of (probabilistic) structured classifiers stochastically approximates the gradient of the objective, but does not approximate the objective itself. Hence, for learning with CD usually Stochastic Gradient Descent (SGD) (Robbins and Monro 1951; Kiefer and Wolfowitz 1952) is used, which is also well-suited for large-scale training. The gradient of the log-likelihood loss is given in (2.19) and consists of two terms:

stochastic gradient descent

$$\frac{\partial \ell_{ll}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{w}} = -\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) + \mathbb{E}_{P(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{w})}[\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}')].$$

The first term, the sufficient statistic of the observed training data is trivial to compute and fixed throughout learning. Whereas, the second term, the expected sufficient statistics by the model (for the current parameters), is generally intractable. CD approximates this term by running an MCMC sampling algorithm, such as Gibbs sampling, for only a few iterations (often only one iteration). The key idea of contrastive divergence is to *initialize the sampling algorithm with the ground-truth observation*. As it is standard for SGD approaches, a learning rate has to be specified. In practice CD has lead to satisfactory results and generally is also fairly efficient (Schmidt, Gao, and Roth 2010).

The convergence of CD is analyzed in (Yuille 2004), where sufficient conditions for its convergence are introduced. Furthermore, CD is related to classical stochastic approximation literature. However, there exist also negative results about the convergence of CD: (Sutskever and Tieleman 2010) shows that (non-regularized) CD updates can not be related to the gradient of any function. Also, an example is shown

where CD cycles indefinitely. All these examples are in the context of restricted Boltzmann machines.

The connections between CD and pseudolikelihood and composite likelihood are studied in (Asuncion et al. 2010). In particular, a blocked CD algorithm is given that performs sampling for a larger subset of variables (a block), than in the standard Gibbs sampling, which only considers one individual variable. This particular algorithm is related to composite likelihood where the likelihood is defined over the same block. Finally, (Vickrey, Lin, and Koller 2010) formulates a *non-local contrastive divergence objective*, which is investigated in more detail in Chapter 5.

2.8.3 *Approximate Maximum Margin Learning*

Approximate learning with the maximum margin surrogate loss is studied in (Finley and Joachims 2008). One particularly interesting aspect of their work, is the fact that they found structured SVMs to work well in combination with LP relaxations. The constraint generation method in the cutting-plane algorithm in their work is also based on LP relaxations. For fractional LP solutions, instead of rounding to valid solutions, Finley and Joachims (2008) generate the constraints directly based on the fractional solutions, which can be achieved by weighting the feature maps and losses of the different outcomes according to the marginal probability of the solution. Learning with outer bounds is particularly attractive, as it makes learning harder than it actually is, if the true marginal polytope would be used.

3

LPQP FOR MPE INFERENCE

In the present chapter we investigate the problem of Most Probable Explanation (MPE) inference in settings where the Linear Program (LP) relaxation is not tight. Our approach is based on augmenting the convex LP relaxation by a non-convex term that penalizes inconsistencies present in the LP formulation. A scalar parameter that can be understood as a temperature, gradually increases the importance of this penalty term. A similar idea will also be used in Chapter 5 for learning CRFs.

3.1 INTRODUCTION

We study the problem of MPE or equivalently Maximum-A-Posteriori (MAP) inference in graphical models. As introduced in Section 2.7, the MPE task is to compute a minimum energy assignment of a set of dependent variables. In the general case, MPE inference is intractable, and therefore most of the current research efforts are concentrated on finding efficient and accurate approximation algorithms. In recent years, Linear Program (LP) relaxations (see Subsection 2.7.4) gained popularity due to their proven success in relevant applications. Several efficient algorithms have been developed to solve the linear program emerging from the relaxation. Despite their success, in many practical problems the solution attained by the LP relaxations is still far from the global minimum.

Our work improves over the LP relaxation by leveraging on a second class of relaxations, namely the Quadratic Program (QP) relaxation (see Subsection 2.7.5). The QP formulation offers a concise and compact description of the MPE problem. We formulate a joint LP and QP MPE objective, that encourages auxiliary variables present in the LP relaxation, to agree with their counterpart in the QP relaxation, through a penalty function. Despite of the non-convexity of this objective, we show that by slowly increasing the weight of the penalty, the solutions found are either competitive with, or in most cases better than the LP

relaxation solutions. This is in general not the case for the few existing QP relaxation solvers.

We propose two variants of the penalty function, each leading to a different Linear and Quadratic Program relaxation (LPQP) objective. We show that the resulting non-convex objectives can be decomposed into a difference of convex functions, which we solve using the Concave-Convex Procedure (CCCP). Having mastered the non-convexity with the CCCP, we solve one of the remaining convex problems with the dual decomposition method, and show that the other can be addressed with the norm-product belief propagation. Interestingly, the main computational task of both of the resulting LPQP algorithms, turns out to be solving known entropy-augmented LPs.

Our contributions are as follows: First we introduce a combined LPQP objective, incorporating the QP constraints through a soft penalty function in the objective. We propose two alternatives for the penalty function, which differ in the way the edges in the graph are weighted. Secondly, we derive CCCP based algorithms for the LPQP objectives, and show that their core computational effort reduces to current entropy-augmented LP solvers. This demonstrates that these modern LP solvers can in some cases be utilized in a smarter way than simply solving the original LP relaxation, leading to possibly faster convergence, as well as lower energy MPE solutions. Through experiments on various datasets, we demonstrate the performance of the suggested LPQP MPE inference in comparison to other commonly used solvers.

3.2 BACKGROUND AND NOTATION

For the sake of clarity we focus on pairwise graphical models. The extension to higher-order factors is relatively straightforward, at the price of a more complicated notation and increased complexity due to the larger factors. Also, we assume that the *energy function is specified explicitly through unary and pairwise potentials*, rather than implicitly through parameters and feature functions. Therefore we can state the MPE problem as follows: For an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ assign each node in the graph to a class or category, such that the overall assignment minimizes an associated energy. Let y_i denote a discrete

variable with a finite domain $\mathcal{Y}_i$, with $|\mathcal{Y}_i| = K$[1], representing the assignment of the i-th node. The MPE problem is defined as

$$\min_{\boldsymbol{y}} \sum_{i \in \mathcal{V}} \theta_i(y_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(y_i, y_j). \tag{3.1}$$

Where $\theta_i(y_i)$ and $\theta_{ij}(y_i, y_j)$ are unary and pairwise potential functions associated with the node and edge assignments. Problem (3.1) can be expressed as an integer quadratic program using a K-ary coding:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \sum_{i \in \mathcal{V}} \boldsymbol{\theta}_i^\top \boldsymbol{\mu}_i + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\mu}_i^\top \boldsymbol{\Theta}_{ij} \boldsymbol{\mu}_j \\ \text{s.t.} \quad & \mu_{i;k} \in \{0,1\} \quad \forall i,k \quad \text{and} \quad \sum_k \mu_{i;k} = 1 \quad \forall i. \end{aligned} \tag{3.2}$$

The pairwise and unary potentials in (3.2), are represented as a matrix $\boldsymbol{\Theta}_{ij}$ and a vector $\boldsymbol{\theta}_i$, respectively.

Variational approaches (see Subsection 2.7.3) to MPE inference reformulate the combinatorial optimization problem in (3.1) as a continuous optimization problem. The next sections formally define two such approaches, namely the LP and QP relaxations. In general, the LP minimization results in a *lower bound* on the energy of the global minimizer, while the QP results in an *upper bound*. The constraint set of the two relaxations is illustrated in Figure 3.1.

3.2.1 *Linear Programming Relaxation*

The LP approach (Schlesinger 1976; Wainwright and Jordan 2008) is based on a convex relaxation of (3.2), where an additional variable $\boldsymbol{\mu}_{ij}$ is included for each edge. Proper *local* marginalization is enforced through summation constraints. Let us define $\boldsymbol{\theta}$ as the vectorized version of the matrix $\boldsymbol{\Theta}$, i.e. $\boldsymbol{\theta} = \text{vec}(\boldsymbol{\Theta})$ The LP reads as

$$\min_{\boldsymbol{\mu} \in \mathcal{L}_\mathcal{G}} \sum_{i \in \mathcal{V}} \boldsymbol{\theta}_i^\top \boldsymbol{\mu}_i + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\theta}_{ij}^\top \boldsymbol{\mu}_{ij}, \tag{3.3}$$

with $\mathcal{L}_\mathcal{G}$, the local marginal polytope:

$$\mathcal{L}_\mathcal{G} = \left\{ \boldsymbol{\mu} \;\middle|\; \begin{array}{ll} \sum_k \mu_{i;k} = 1 & \forall i \in \mathcal{V} \\ \sum_l \mu_{ij;kl} = \mu_{i;k} & \forall k, (i,j) \in \mathcal{E} \\ \sum_k \mu_{ij;kl} = \mu_{j;l} & \forall l, (i,j) \in \mathcal{E} \\ \mu_{ij;kl} \geq 0 & \forall k,l, (i,j) \in \mathcal{E} \end{array} \right\}.$$

1 For notational convenience we assume $\mathcal{Y}_i = \{1, \ldots, K\}$; in the experiments we will however also consider settings where the domain of the variables has different size.

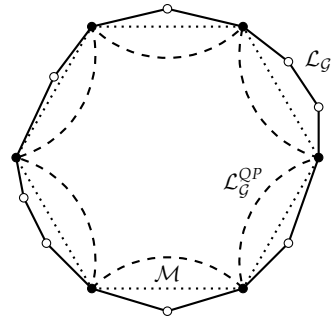


Figure 3.1: Schematic illustration of the constraint set in the LP (solid) and the QP (dashed) relaxations. While the LP relaxation is an outer bound on the true marginal polytope (dotted) leading to additional fractional solutions (shown as non-filled circles), the QP relaxation gives an inner bound. We use $\mathcal{L}_\mathcal{G}^{QP}$ to denote the local marginal polytope with additional quadratic consistency constraints.

In the general case, $\mathcal{L}_\mathcal{G}$ is an inexact description of the marginal polytope $\mathcal{M}$, which requires an exponentially large number of constraints (Wainwright and Jordan 2008). If $\mathcal{L}_\mathcal{G}$ in (3.3) is replaced by $\mathcal{M}$, then the solution recovers the true MPE assignment. A solution to an LP-based approach (for a constraint set consisting of any subset of the marginal polytope, such as $\mathcal{L}_\mathcal{G}$), admits an easy to verify certificate of optimality. If the solution is integer, it is the global optimum.

The work in (Sontag et al. 2008) proposes to tighten the polytope by including summation constraints over larger subsets of variables. This approach has been successful in identifying the global minimum for some problems. However, it suffers from an increased complexity as ultimately an exponentially large set of possible constraints might need to be searched over. In practice a class of possible summation constraints are considered, such as those consisting of all triplets.

3.2.2 *Quadratic Programming Relaxation*

An alternative relaxation of the integer quadratic program in (3.2) is obtained by simply dropping the integer constraints. The resulting QP is given by:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \sum_{i \in \mathcal{V}} \boldsymbol{\theta}_i^{\mathsf{T}} \boldsymbol{\mu}_i + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\Theta}_{ij} \boldsymbol{\mu}_j \\ \text{s.t.} \quad & 0 \leq \mu_{i;k} \leq 1 \quad \forall i, k \quad \text{and} \quad \sum_k \mu_{i;k} = 1 \quad \forall i. \end{aligned} \tag{3.4}$$

A major advantage of the QP relaxation, is its tightness, which means that the minimizer of (3.4) also minimizes (3.1), as was shown in (Ravikumar and Lafferty 2006). The QP also benefits from a more compact description compared to the LP relaxation, as it requires fewer constraints and variables to formulate the *exact* MPE problem. The variable vector $\boldsymbol{\mu}$, is of size $K \cdot |\mathcal{V}| + K^2 \cdot |\mathcal{E}|$ in the LP (3.3) and only $K \cdot |\mathcal{V}|$ in the QP. The biggest drawback of the QP relaxation, is that in the general case the optimization problem turns out to be non-convex due to the edges product term. This fact renders an exact minimization difficult. A QP solution is not necessarily guaranteed to be integer. As was shown in (Ravikumar and Lafferty 2006), a (local) solution to the QP relaxation can always be rounded to an integer solution with smaller or equal energy. We will discuss this rounding strategy in Subsection 3.4.2.

In terms of motivation, our work is similar to the QP relaxation approach. The QP formulation of the MPE problem (3.4) was introduced in (Ravikumar and Lafferty 2006), but stems from classical mean-field approaches. Ravikumar and Lafferty (2006) solved the non-convex problem using a convex relaxation. The solution was later improved in (Kappes and Schnörr 2008) through a difference of convex functions formulation. Both solvers are generic in the sense that they do not exploit the graph structure. Recently (Kumar and Zilberstein 2011) introduced a message-passing algorithm for solving the QP relaxation. While improving the run time over the other two algorithms, it still generally suffers from poor solutions due to local minima. The QP solvers often deal with this drawback by restarting with different initializations. We observed that our LPQP algorithms, are much more resilient with respect to the initialization. In all of the experiments we conducted, a restart was never required. We attribute this behavior to the gradual progression between the LP and QP. Finally, in concurrent

work (Kumar, Zilberstein, and Toussaint 2012) propose a hybrid LP and QP approach to MPE, similar to our formulation discussed in the next section. The resulting optimization problem is solved by a custom message-passing scheme. Our work on the other hand, in its essence reduces to well-known entropy-augmented LP objectives, for which efficient message-passing algorithms exist.

3.3 COMBINED LP AND QP RELAXATION

We propose to optimize an objective which is a combination of the LP and QP relaxations. We retain the auxiliary variables $\boldsymbol{\mu}_{ij}$ of the pairwise terms, but force these variables to agree with the product of the unary marginals $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$. The constraints, given by $\text{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T}) = \boldsymbol{\mu}_{ij}\ \forall (i,j) \in \mathcal{E}^2$, are enforced through a penalty function $g(\cdot)$ incorporated in the objective. The extent to which the constraint is enforced, is regulated by the parameter ρ.

We focus on the Kullback-Leibler (KL) divergence as the penalty function. In our setting this is equivalent to the mutual information between the unary and marginal variables. The choice of the penalty function is motivated by the probabilistic nature of the compared marginal terms. Moreover, as we will see later, the use of the KL divergence as a penalty term gives rise to efficient message-passing algorithms for the solution of the resulting optimization problems. As a reminder, for probability distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ of a discrete random variable, their KL divergence is defined to be

$$D_{KL}(\boldsymbol{\mu}, \boldsymbol{\nu}) := \sum_k \mu_k \log\left(\frac{\mu_k}{\nu_k}\right).$$

Our approach enforces consistency between the unary and pairwise marginals for each edge using the KL divergence. As these marginals are properly normalized, which is ensured through the constraints in the local marginal polytope, the KL divergence simply corresponds to the mutual information I of the marginals for an edge (i,j), see (2.31):

$$I_{ij}(\boldsymbol{\mu}_{ij}, \text{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T})) = H(\boldsymbol{\mu}_i) + H(\boldsymbol{\mu}_j) - H(\boldsymbol{\mu}_{ij}).$$

In our work the mutual information between the pairwise term $\boldsymbol{\mu}_{ij}$ and the product of unary terms $\text{vec}(\boldsymbol{\mu}\boldsymbol{\mu}^\mathsf{T})$ is *minimized*, as in the consistent

2 Here $\text{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T})$ denotes the vectorized version of the outer product of $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$.

case the mutual information is zero. The general form of the combined LPQP objective reads as

$$\min_{\boldsymbol{\mu} \in \mathcal{L}_\mathcal{G}} \boldsymbol{\theta}^\mathsf{T} \boldsymbol{\mu} + \rho g(\boldsymbol{\mu}). \tag{3.5}$$

The first term is simply the LP objective (3.3), written as a scalar product between the potential function, and the concatenated unary and pairwise variables. We investigate two constructions of the penalty term $g(\boldsymbol{\mu})$. The constructions differ in the weighting of the edges. The penalty function has the property that it is positive and only zero if the unary marginals agree with the pairwise marginals for all the edges. For $\rho = 0$, (3.5) amounts to the standard LP relaxation. On the other extreme when $\rho \to \infty$, the constraints $\mathrm{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T}) = \boldsymbol{\mu}_{ij}\ \forall (i,j) \in \mathcal{E}$ are fulfilled and the QP relaxation is recovered. By successively increasing ρ during the run of our algorithms, we achieve a gradual enforcement of the constraints.

UNIFORM WEIGHTING The KL divergence is penalized in the same way for all the edges in the graph:

$$\begin{aligned} g^{uni}(\boldsymbol{\mu}) &:= \sum_{(i,j)\in\mathcal{E}} D_{KL}(\boldsymbol{\mu}_{ij}, \mathrm{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T})) \\ &= \sum_{(i,j)\in\mathcal{E}} I_{ij}(\boldsymbol{\mu}_{ij}, \mathrm{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T})) \\ &= \sum_{i\in\mathcal{V}} d_i H(\boldsymbol{\mu}_i) - \sum_{(i,j)\in\mathcal{E}} H(\boldsymbol{\mu}_{ij}). \end{aligned} \tag{3.6}$$

As before, d_i denotes the degree of the i-th node.

TREE-BASED WEIGHTING Let $\mathcal{A}$ denote a set of trees. In the tree-based weighting, the KL divergence is penalized uniformly within a forest-shaped subgraph:

$$\begin{aligned} g^{tree}(\boldsymbol{\mu}) &:= \sum_{a\in\mathcal{A}} \eta_a \left(\sum_{(i,j)\in\mathcal{E}_a} D_{KL}(\boldsymbol{\mu}_{ij}, \mathrm{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T})) \right) \\ &= \sum_{a\in\mathcal{A}} \eta_a \left(\sum_{(i,j)\in\mathcal{E}_a} I_{ij}(\boldsymbol{\mu}_{ij}, \mathrm{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T})) \right) \\ &= \sum_{a\in\mathcal{A}} \eta_a \left(\sum_{i\in\mathcal{V}_a} d_i^a H(\boldsymbol{\mu}_i) - \sum_{(i,j)\in\mathcal{E}_a} H(\boldsymbol{\mu}_{ij}) \right). \end{aligned} \tag{3.7}$$

We assume that a decomposition of the original graph into acyclic subgraphs (e.g. trees or individual edges) exists, and is given by

$$\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a), \qquad \mathcal{V} = \bigcup_{a \in \mathcal{A}} \mathcal{V}_a, \qquad \mathcal{E} = \bigcup_{a \in \mathcal{A}} \mathcal{E}_a.$$

The positive weights η_a are tree specific, and assumed to sum to one. In this work we simply used $\eta_a = 1/|\mathcal{A}|$. Figure 3.2 visualizes two different choices of acyclic decompositions for a regular grid.

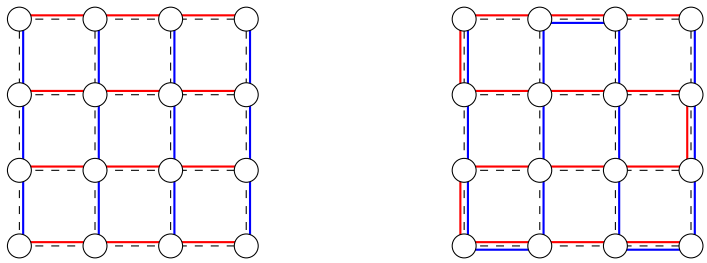


Figure 3.2: Two different decompositions of a regular grid. The choice on the left includes each edge in only one of the two decompositions, whereas the choice on the right covers some edges on the boundary twice. Both decompositions are valid in our framework, but might however lead to slightly different convergence properties or identify different local minima of the QP objective. Jancsary and Matz (2011) refer to the decomposition on the right as "snakes". For regular grid graphs we usually use the decomposition on the left.

DIFFERENCE TO THE BETHE FREE ENERGY We would like to contrast the LPQP approach to a popular objective for *marginal* inference (as opposed to MPE inference), the *Bethe free energy*. As discussed in Subsection 2.7.3, marginal inference using variational methods has the additional problem, that in general the entropy does not factorize into individual marginals and therefore approximations to the entropy have to be found. One popular entropy approximation is given by the Bethe entropy, see (Wainwright and Jordan 2008, Chapter 4.1):

$$H(P) \approx H_{\text{Bethe}}(\boldsymbol{\mu}) = \sum_{i \in \mathcal{V}} H(\boldsymbol{\mu}_i) - \sum_{(i,j) \in \mathcal{E}} I_{ij}(\boldsymbol{\mu}_{ij}, \operatorname{vec}(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^\mathsf{T})). \tag{3.8}$$

This is exact in case the graphical model is a tree, and an approximation in the general case. Comparing the Bethe entropy approximation

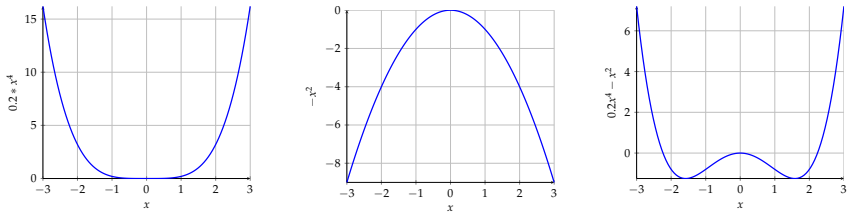
in (3.8) and the mutual information penalty term in (3.6), we notice[3] that the only difference between the two objectives is the unary entropy. Hence the LPQP objective can also be understood as a Bethe free energy, without the part that encourages configurations with a large entropy. As we are ultimately interested in the MPE, which is integer, an entropy term is not desirable.

3.4 LPQP ALGORITHMS

In this section we derive two algorithms for the non-convex LPQP objective in (3.5), with the different penalty terms in (3.6) and (3.7).

3.4.1 *Difference of Convex Functions*

The Concave-Convex Procedure (CCCP) (Yuille and Rangarajan 2003), can be applied to a constrained optimization problem, where the objective is non-convex, provided that the objective has a decomposition into a convex and a concave part, see Figure 3.3. CCCP has already been applied to inference problems in the context of the Bethe free energy (Yuille 2002) and recently to solve the QP relaxation (Kumar and Zilberstein 2011; Kumar, Zilberstein, and Toussaint 2012; Kappes and Schnörr 2008).



(a) Convex function. (b) Concave function. (c) DC function.

Figure 3.3: A DC function consists of a convex and a concave part.

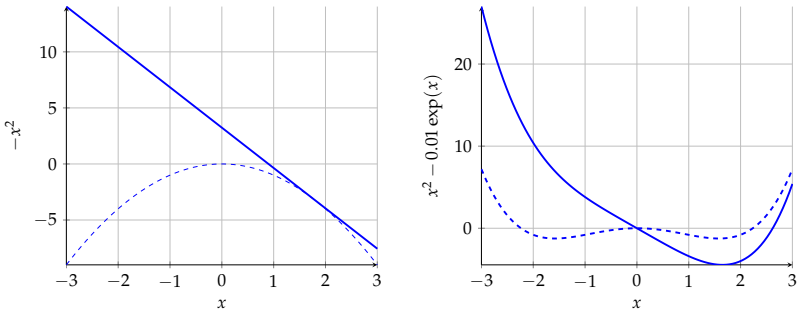
In our setting, we wish to find a decomposition of the form

$$\min_{\boldsymbol{\mu} \in \mathcal{L}_{\mathcal{G}}} u_\rho(\boldsymbol{\mu}) - v_\rho(\boldsymbol{\mu}),$$

3 Note that we are minimizing the *negative* entropy in a variational principle

where both, $u_\rho(\boldsymbol{\mu})$ and $v_\rho(\boldsymbol{\mu})$ are convex. The CCCP algorithm proceeds by iteratively solving a convexified objective, obtained by a linearization of $v_\rho(\boldsymbol{\mu})$, see Figure 3.4:

$$\boldsymbol{\mu}^{t+1} = \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{L}_\mathcal{G}} u_\rho(\boldsymbol{\mu}) - \boldsymbol{\mu}^\mathsf{T} \nabla v_\rho(\boldsymbol{\mu}^t). \tag{3.9}$$



(a) Approximation of the concave function with a linear function.

(b) Convexified approximation of the DC function.

Figure 3.4: The concave part of a DC function is approximated by a linear function. We show the convexified objective as a solid line and the original objective as a dashed line.

The decompositions of the two LPQP objectives, as well as the gradients of the concave part, are shown below. In the derivations we used the definition of the mutual information in terms of entropies, which holds due to the marginalization constraints of the pairwise marginals (Wainwright and Jordan 2008).

For both objectives, the convex part $u_\rho(\boldsymbol{\mu})$ consists of the original LP formulation, with an additional term that encourages configurations with a large entropy. In the uniform weights penalty, this additional term takes the form of the entropy of the pairwise marginals, whereas in the tree-based penalty, it constitutes of the sum of tree entropies.

The concave part of the decompositions, v_ρ, corresponds to an entropy of the unary marginals. In the CCCP step (3.9), $\log(\boldsymbol{\mu}_i)$ is replaced by $\log(\boldsymbol{\mu}_i^t)$, the marginal from the previous iteration, resulting in an entropy approximation.

UNIFORM WEIGHTING The difference of convex function decomposition of the combined LPQP objective in (3.5) for the uniform penalty term in (3.6) is given by:

$$u_\rho(\boldsymbol{\mu}) = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\mu} - \rho \sum_{(i,j)\in\mathcal{E}} H(\boldsymbol{\mu}_{ij})$$
$$v_\rho(\boldsymbol{\mu}) = -\rho \sum_{i\in\mathcal{V}} d_i H(\boldsymbol{\mu}_i).$$

Here d_i denotes the degree of the i-th node in the graph. The derivative of the concave part w.r.t. the unary marginals is

$$\frac{\partial v_\rho(\boldsymbol{\mu})}{\partial \mu_{i;k}} = \rho d_i (1 + \log \mu_{i;k}),$$

whereas the derivative w.r.t. the pairwise marginals is zero.

TREE-BASED WEIGHTING The difference of convex function decomposition of the combined LPQP objective in (3.5) for the tree weighted penalty term in (3.7) can be achieved as follows:

$$u_\rho(\boldsymbol{\mu}) = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\mu} - \rho \sum_{a\in\mathcal{A}} \eta_a \left(\sum_{(i,j)\in\mathcal{E}_a} H(\boldsymbol{\mu}_{ij}) - \sum_{i\in\mathcal{V}_a} (d_i^a - 1) H(\boldsymbol{\mu}_i) \right)$$
$$v_\rho(\boldsymbol{\mu}) = -\rho \sum_{a\in\mathcal{A}} \eta_a \sum_{i\in\mathcal{V}_a} H(\boldsymbol{\mu}_i).$$

Here d_i^a denotes the degree of the i-th node in the subgraph indexed by a. $\mathcal{A}(i)$ denotes the set of all trees that contain node i. The derivative of the concave part w.r.t. the unary marginals is

$$\frac{\partial v_\rho(\boldsymbol{\mu})}{\partial \mu_{i;k}} = \rho \sum_{a\in\mathcal{A}(i)} \eta_a (1 + \log \mu_{i;k}).$$

As in the uniform case, the derivative of the concave part w.r.t. to the pairwise marginals is zero.

SUMMARY The LPQP objective consists of the standard LP relaxation with an additional penalty term that encourages consistencies between the unary and pairwise marginal variables. The resulting non-convex objective is decomposed into a difference of convex functions to which CCCP is applied. The algorithm reduces to solving a convex minimization problem, which consists of a linear part and unary and pairwise

negative entropy terms. The CCCP step can be understood as a modification of the unary potentials $\boldsymbol{\theta}_i$ based on the solution in the previous iteration.

3.4.2 *Algorithm Overview*

The general scheme of the suggested LPQP algorithms is shown in Algorithm 3.1. The algorithm consists of two loops. The inner loop solves the DC problem for a fixed penalty parameter ρ, whereas the outer loop gradually increases the value of ρ. Increasing ρ implies that the penalty function for the quadratic constraint contributes more to the overall objective, eventually resulting in a QP solution for which all the constraints are fulfilled.

Algorithm 3.1 LPQP algorithm scheme for MPE.

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E}), \boldsymbol{\theta}, \rho_0$.
1: initialize $\boldsymbol{\mu} \in \mathcal{L}_\mathcal{G}$ uniform, $\rho = \rho_0$.
2: **repeat**
3: $\quad t = 0, \boldsymbol{\mu}^0 = \boldsymbol{\mu}$.
4: $\quad$ **repeat**
5: $\quad\quad \boldsymbol{\mu}^{t+1} = \operatorname{argmin}_{\boldsymbol{\tau} \in \mathcal{L}_\mathcal{G}} u_\rho(\boldsymbol{\tau}) - \boldsymbol{\tau}^\mathsf{T} \nabla v_\rho(\boldsymbol{\mu}^t)$.
6: $\quad\quad t = t + 1$.
7: $\quad$ **until** $\|\boldsymbol{\mu}^t - \boldsymbol{\mu}^{t-1}\|_2 \leq \epsilon_{\text{dc}}$.
8: $\quad \boldsymbol{\mu} = \boldsymbol{\mu}^t$.
9: $\quad$ increase ρ.
10: **until** $\|\boldsymbol{\mu} - \boldsymbol{\mu}^0\|_2 \leq \epsilon_\rho$.
11: **return** $\boldsymbol{\mu}$.

The main computational task is in line 5, where a particular instance of a convex optimization problem is solved. Subsection 3.4.3 and Subsection 3.4.4 discuss efficient algorithms for solving this optimization problem for the two different weighting schemes. In this thesis we choose two different types of algorithms for the solution of the problem: A coordinate-descent algorithm for the uniform weighting case and a gradient-descent algorithm for the tree weighting case. The methods are likely to perform well in the corresponding setting. However, it is very likely that a gradient based algorithm could also be used for the uniform weighting, and vice versa a coordinate-descent algorithm for the tree weighted case.

Warm-starting the problem in line 5 with the previous solution between successive calls, leads to a substantial speed-up. We choose the initial $\rho = \rho_0$ depending on the scaling of the energies, and use a multiplicative increase with a fixed value. In the experiments we use a multiplicative factor of 1.5, but the results were not very sensitive to this choice.

SOLUTION ROUNDING Similarly to the LP and QP relaxations, the solutions returned by the LPQP algorithms can be fractional. Since the LPQP scheme ultimately solves a variant of the QP relaxation, to attain the final integer solutions, we use the QP solution rounding scheme suggested in (Ravikumar and Lafferty 2006). Given unary marginals $\boldsymbol{\mu}^*$, we assign the i-th node the label y_i^* given by

$$y_i^* = \operatorname*{argmin}_k \left(\theta_{i;k} + \sum_{j \in \mathcal{N}(i)} \sum_l \theta_{i,j;k,l} \mu_{j;l}^* \right).$$

Here $\mathcal{N}(i)$ denotes the neighbors of node i. After determining the label of the i-th variable, we set $\mu_{i;y_i^*}^* = 1$ and $\mu_{i;k}^* = 0 \quad \forall\, k \neq y_i^*$, and continue until labels are assigned to all nodes. It can be verified that the rounded solution has an energy that is smaller or equal to the energy of the initial solution, see (Ravikumar and Lafferty 2006).

3.4.3 *Uniform Weighting*

The convex sub-problem we get in the CCCP step with the uniform weighting penalty function (3.6), is given by

$$\min_{\boldsymbol{\mu} \in \mathcal{L}_\mathcal{G}} \sum_{i \in \mathcal{V}} \tilde{\boldsymbol{\theta}}_i^\mathsf{T} \boldsymbol{\mu}_i + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\theta}_{ij}^\mathsf{T} \boldsymbol{\mu}_{ij} - \rho \sum_{(i,j) \in \mathcal{E}} H(\boldsymbol{\mu}_{ij}). \tag{3.10}$$

where $\tilde{\boldsymbol{\theta}}_i$, is a modification of the unary potentials by an additional gradient term, originating in the linearized part of the DC decomposition (3.9) [4]:

$$\tilde{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_i - \rho d_i \log(\boldsymbol{\mu}_i^t). \tag{3.11}$$

As a result of this unary potentials modification, configurations with small probability in the previous iteration t, are vigorously dis-encouraged.

4 The ρd_i term in ∇v_ρ is constant and can therefore be dropped.

BELIEF PROPAGATION The convex problem in (3.10) is solved by the norm-product Belief Propagation (BP) (Hazan and Shashua 2010). This algorithm is a generalization of belief-propagation (Yedidia, Freeman, and Weiss 2005) and tree-reweighted BP (Wainwright, Jaakkola, and Willsky 2005b). It is a primal-dual ascent algorithm and is guaranteed to converge to the global optimum for any choice of $\rho > 0$.

The norm-product algorithm applied to (3.10) computes messages passed from node j to node i as follows

$$m_{j\to i}(y_i) \propto \left(\sum_{y_j} \psi_{ij}^{1/\rho}(y_i, y_j) \frac{\psi_j^{1/(d_j\rho)}(y_j) \prod_{s\in\mathcal{N}(j)} m_{s\to j}^{1/(d_j\rho)}(y_j)}{m_{i\to j}^{1/\rho}(y_j)} \right)^{\rho},$$

where we choose to define $\psi_{ij}(y_i, y_j) := \exp(-\theta_{ij}(y_i, y_j))$ and $\psi_i(y_i) := \exp(-\tilde{\theta}_i(y_i))$. Upon convergence the marginals $\boldsymbol{\mu}_i$ are obtained by multiplying the incoming messages at variable i:

$$\mu_i(y_i) \propto \left(\psi_i(y_i) \prod_{j\in\mathcal{N}(i)} m_{j\to i}(y_i) \right)^{1/(d_i\rho)}.$$

Due to warm starting with the previous CCCP iteration solution, typically only few passes through the graph are needed for the messages to converge in the later stages of the run.

3.4.4 *Tree-based Weighting*

The convex sub-problem corresponding to the CCCP step with the tree-based weighting penalty (3.7) is,

$$\min_{\boldsymbol{\mu}\in\mathcal{L}_\mathcal{G}} \sum_{i\in\mathcal{V}} \tilde{\boldsymbol{\theta}}_i^\mathsf{T} \boldsymbol{\mu}_i + \sum_{(i,j)\in\mathcal{E}} \boldsymbol{\theta}_{ij}^\mathsf{T} \boldsymbol{\mu}_{ij} - \rho \sum_{a\in\mathcal{A}} \eta_a H_{\text{tree}}^a(\boldsymbol{\mu}). \tag{3.12}$$

Here we define the entropy of a tree by

$$H_{\text{tree}}^a(\boldsymbol{\mu}) := \left(\sum_{(i,j)\in\mathcal{E}_a} H(\boldsymbol{\mu}_{ij}) - \sum_{i\in\mathcal{V}_a} (d_i^a - 1) H(\boldsymbol{\mu}_i) \right).$$

As before, the linearization of the concave part in the CCCP step, results in a modification of the unaries

$$\tilde{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_i - \rho \sum_{a\in\mathcal{A}(i)} \eta_a \log(\boldsymbol{\mu}_i^t). \tag{3.13}$$

Below we illustrate how dual decomposition can be used to derive a message-passing algorithm for the minimization in (3.12).

DUAL DECOMPOSITION The dual decomposition framework (Bertsekas 1999; Komodakis, Paragios, and Tziritas 2007), can be applied to an optimization problem provided that the objective can be decomposed into several sub-problems, also known in the literature as the slave problems. The global variables, $\boldsymbol{\mu}$ in our case, are replaced with local copies in each slave problem, denoted here by $\boldsymbol{\nu}^a$, such that the minimization of the slave problems can be carried out independently. To enforce the local variables corresponding to the same original variables to assume the same value, a designated constraint is introduced. The optimization of the sum of slave problems, subject to these constraints, is called the master problem. A dual decomposition of problem (3.12), was carried out in (Domke 2011). We use the same decomposition, but take a different route optimizing the resulting master problem.

$$\begin{aligned} \min_{\boldsymbol{\mu} \in \mathcal{L}_\mathcal{G}} \quad & \sum_{a \in \mathcal{A}} \min_{\boldsymbol{\nu}^a \in \mathcal{L}_{\mathcal{G}_a}} s_a(\boldsymbol{\nu}^a) \\ \text{s.t.} \quad & \boldsymbol{\nu}^a_i = \boldsymbol{\mu}_i \quad \forall i, a \in \mathcal{A}. \\ & \boldsymbol{\nu}^a_{ij} = \boldsymbol{\mu}_{ij} \quad \forall (i,j), a \in \mathcal{A}. \end{aligned} \tag{3.14}$$

Here the slave problems are defined as

$$s_a(\boldsymbol{\nu}) := \sum_{i \in \mathcal{V}_a} \hat{\boldsymbol{\theta}}_i^\mathsf{T} \boldsymbol{\nu}_i + \sum_{(i,j) \in \mathcal{E}_a} \hat{\boldsymbol{\theta}}_{ij}^\mathsf{T} \boldsymbol{\nu}_{ij} - \rho \eta_a H^a_{\text{tree}}(\boldsymbol{\nu}).$$

Note that since the summation over the trees now extends to include the unary and pairwise terms, the corresponding potentials should be adjusted accordingly

$$\hat{\boldsymbol{\theta}}_i = \frac{\tilde{\boldsymbol{\theta}}_i}{|\mathcal{A}(i)|}, \qquad \hat{\boldsymbol{\theta}}_{ij} = \frac{\boldsymbol{\theta}_{ij}}{|\mathcal{A}(i,j)|}.$$

Each slave problem is defined over a tree structured graph and can therefore be solved exactly using the sum-product algorithm, in two passes over the tree. The temperature in this case is $\rho \eta_a$.

In practice, instead of (3.14), we consider the following master problem

$$\begin{aligned} & \sum_{a \in \mathcal{A}} \min_{\boldsymbol{\nu}^a \in \mathcal{L}_{\mathcal{G}_a}} s_a(\boldsymbol{\nu}^a) \\ \text{s.t.} \quad & \boldsymbol{\nu}_i^a = \frac{1}{|\mathcal{A}(i)|} \sum_{a' \in \mathcal{A}(i)} \boldsymbol{\nu}_i^{a'} \quad \forall i, a \in \mathcal{A}(i) \\ & \boldsymbol{\nu}_{ij}^a = \frac{1}{|\mathcal{A}(i,j)|} \sum_{a' \in \mathcal{A}(i,j)} \boldsymbol{\nu}_{ij}^{a'} \quad \forall (i,j), a \in \mathcal{A}(i,j). \end{aligned} \tag{3.15}$$

Here we use the idea from (Domke 2011) who formulates the constraint on the replicated marginal variables to agree with the mean. This is simpler than the constraint in (3.14), as one does not need to introduce the additional variable $\boldsymbol{\mu}$. We can write the Lagrangian and rearrange to get

$$\mathcal{L}(\boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{|\mathcal{A}|}, \boldsymbol{\lambda}) = \sum_{a \in \mathcal{A}} \min_{\boldsymbol{\nu}^a \in \mathcal{L}_{\mathcal{G}_a}} \left(s_a(\boldsymbol{\nu}^a) + \sum_{i \in \mathcal{V}_a} \boldsymbol{\theta}_i^a(\boldsymbol{\lambda})^\mathsf{T} \boldsymbol{\nu}_i^a + \sum_{(i,j) \in \mathcal{E}_a} \boldsymbol{\theta}_{ij}^a(\boldsymbol{\lambda})^\mathsf{T} \boldsymbol{\nu}_{ij}^a \right),$$

with

$$\begin{aligned} \boldsymbol{\theta}_i^a(\boldsymbol{\lambda}) &= \boldsymbol{\lambda}_i^a - \frac{1}{|\mathcal{A}(i)|} \sum_{a' \in \mathcal{A}(i)} \boldsymbol{\lambda}_i^{a'} \\ \boldsymbol{\theta}_{ij}^a(\boldsymbol{\lambda}) &= \boldsymbol{\lambda}_{ij}^a - \frac{1}{|\mathcal{A}(i,j)|} \sum_{a' \in \mathcal{A}(i,j)} \boldsymbol{\lambda}_{ij}^{a'}. \end{aligned}$$

The Lagrange multipliers vector $\boldsymbol{\lambda}$ is of the same length as all the $\boldsymbol{\nu}$ concatenated together, where for variables that are only replicated once, the corresponding Lagrange multiplier can be dropped. We can think of the potentials as being a function of $\boldsymbol{\lambda}$ and thus the dual problem of (3.15) is given by

$$\max_{\boldsymbol{\lambda}} \sum_{a \in \mathcal{A}} \min_{\boldsymbol{\nu}^a \in \mathcal{L}_{\mathcal{G}_a}} s_a(\boldsymbol{\nu}^a, \boldsymbol{\lambda}). \tag{3.16}$$

Here $s_a(\boldsymbol{\nu}^a, \boldsymbol{\lambda})$ is defined by

$$s_a(\boldsymbol{\nu}, \boldsymbol{\lambda}) := \sum_{i \in \mathcal{V}_a} \hat{\boldsymbol{\theta}}_i^a(\boldsymbol{\lambda})^\mathsf{T} \boldsymbol{\nu}_i + \sum_{(i,j) \in \mathcal{E}_a} \hat{\boldsymbol{\theta}}_{ij}^a(\boldsymbol{\lambda})^\mathsf{T} \boldsymbol{\nu}_{ij} - \rho \eta_a H_{\text{tree}}^a(\boldsymbol{\nu}),$$

and the modified potentials are given as:

$$\hat{\boldsymbol{\theta}}_i^a(\boldsymbol{\lambda}) = \hat{\boldsymbol{\theta}}_i + \boldsymbol{\theta}_i^a(\boldsymbol{\lambda})$$
$$\hat{\boldsymbol{\theta}}_{ij}^a(\boldsymbol{\lambda}) = \hat{\boldsymbol{\theta}}_{ij} + \boldsymbol{\theta}_{ij}^a(\boldsymbol{\lambda}).$$

All the slave computations can be carried out exactly using the sum-product algorithm. For the maximization w.r.t. $\boldsymbol{\lambda}$ we use the Fast Iterative Shrinkage-Thresholding (FISTA) algorithm (Beck and Teboulle 2009), a modern variant of Nesterov's traditional fast gradient method (Nesterov 1983). Our dual decomposition approach is very similar to (Savchynskyy et al. 2011), with two key differences. First, the authors study only a specific choice of the decomposition for the 4-connected grid graph in which each node marginal is replicated twice and each edge is only considered once. Second, we are also interested in settings of $\rho \gg 0$, which is not meaningful in the context of the standard LP relaxation.

The algorithm we used for solving the master problem is given in Algorithm 3.2. It is based on FISTA descent as described in (Vandenberghe 2012). In the algorithm $f(\boldsymbol{\lambda})$ denotes the dual objective given in (3.16) for the dual variables $\boldsymbol{\lambda}$.

Algorithm 3.2 FISTA ascent for the master problem in (3.16).

1: initialize $\boldsymbol{\lambda}^{(0)} = \boldsymbol{v}^{(0)} = \mathbf{0}$, $k = 1$.
2: **repeat**
3: $\quad \theta_k = 2/(k+1)$.
4: $\quad \boldsymbol{y} = (1-\theta_k)\boldsymbol{\lambda}^{(k-1)} + \theta_k \boldsymbol{v}^{(k-1)}$.
5: $\quad \boldsymbol{u} = \boldsymbol{y} + t_k \nabla f(\boldsymbol{y})$, perform line-search for t_k.
6: $\quad$ ensure ascent for $\boldsymbol{\lambda}^{(k)}$:

$$\boldsymbol{\lambda}^{(k)} = \begin{cases} \boldsymbol{u} & f(\boldsymbol{u}) \geq f(\boldsymbol{\lambda}^{(k-1)}) \\ \boldsymbol{\lambda}^{(k-1)} & \text{otherwise.} \end{cases}$$

7: $\quad \boldsymbol{v}^{(k)} = \boldsymbol{\lambda}^{(k-1)} + \frac{1}{\theta_k}(\boldsymbol{u} - \boldsymbol{\lambda}^{(k-1)})$.
8: $\quad k = k+1$.
9: **until** converged.
10: **return** $\boldsymbol{\lambda}^{(k-1)}$.

As a stopping criteria we use a minimum gradient norm change condition together with a limit on the number of iterations. The line

search for t_k is given in Algorithm 3.3. The line-search is simpler than in the standard FISTA algorithm, as for the objective in (3.16), no proximal operations are required. The line search increases sufficient advance and guarantees a favorable convergence rate. Alternatively, instead

Algorithm 3.3 The line search used inside the FISTA algorithm.

initialize $0 < \beta < 1$, t_0 any positive value.
$t = t_{k-1}$.
repeat
  $\boldsymbol{u} = \boldsymbol{y} + t\nabla f(\boldsymbol{y})$.
  $f_{sq} = f(\boldsymbol{u}) + \frac{t}{2}\|\nabla f(\boldsymbol{y})\|^2$.
  if $f(\boldsymbol{u}) < f_{sq}$ **then**
    $t = \beta t$.
  end if
until $f(\boldsymbol{u}) \geq f_{sq}$
return t.

of using the accelerated first-order methods described here, it is also possible to directly use L-BFGS for the maximization of the dual objective in (3.16).

3.4.5 *Entropy-augmented LP Solvers*

We would like to contrast the LPQP algorithm to some existing message-passing solvers for the LP relaxation. In practice often Sequential Tree-Reweighted Message Passing (TRWS) (Kolmogorov 2006) or Max-Product Linear Programming (MPLP) (Sontag et al. 2008) are used to solve the LP relaxation. However, these coordinate-descent algorithms all suffer from the problem that they might get stuck and generally do not converge to the minimum of the linear program.

Therefore, recently, several works (Jojic, Gould, and Koller 2010; Savchynskyy et al. 2011) proposed to smooth the LP objective by adding a term that favors entropic marginals. The merit of this additional term is in overcoming the non-smoothness of the objective. In order to ultimately solve the original LP, these entropy-augmented solvers progressively lower the entropy term. Naturally, the convergence of these algorithms is fairly fast in the beginning. This line of research originates in Nesterov's work on fast gradient methods (Nesterov 1983; Nesterov 2005). Another, related line of research is based on adding a quadratic

proximal term to the LP objective. Message-passing algorithms based on the Alternating Direction Method of Multipliers (ADMM) (Boyd et al. 2011) are for example derived in (Martins et al. 2011; Meshi and Globerson 2011). Contrary to the methods based on smoothing, for these augmented Lagrangian approaches, the influence of the proximal term however does not need to be decreased.

The proposed LPQP solvers have the opposite behavior with respect to the smoothness of the objective, which is controlled through ρ. The influence of the entropy term is increased through the progression of the algorithm, leading to favorable convergence properties.

3.4.6 *Convergence of the LPQP Algorithms*

Lemma 3.1. *The concave-convex procedure in Algorithm 3.1 converges to a stationary point of the LPQP objective in (3.5) with $\rho = \rho_{final}$, the parameter value reached when the marginals do not change further.*

Proof. It was shown in (Sriperumbudur and Lanckriet 2009) that the CCCP with a convex constraint set converges to a stationary point of the objective. In the last DC iteration, a CCCP is solved with $\rho = \rho_{\text{final}}$. □

3.5 RELATED WORK

The work in (Kumar, Zilberstein, and Toussaint 2012) is most closely related to our approach. Instead of enforcing the constraint for all the edges, as done in our work, (Kumar, Zilberstein, and Toussaint 2012) suggests to enforce the constraint for only a subset of the edges. For these edges the constraint is however not enforced by a penalty function, but rather by dropping the auxiliary pairwise variables from the objective and replacing them by quadratic terms. Kumar, Zilberstein, and Toussaint (2012) derive a custom message-passing algorithm similar to earlier work in (Kumar and Zilberstein 2011). Their message-passing scheme does however lack the connections to known entropy-augmented LP solvers.

The idea of a gradual enforcement of the constraints through a penalty function is relatively well-known and at least dates back to the work on graduated non-convexity (Blake and Zisserman 1987) for visual reconstruction. The same idea is also used in simulated and

deterministic annealing (Kirkpatrick, Gelatt, and Vecchi 1983; Hofmann and Buhmann 1997) in the context of Gibbs sampling.

3.6 EXPERIMENTS

We use LPQP-U to refer to the implementation of the uniform weighting of the edges, and LPQP-T for the tree-based weighting. In the experiments where the graph did not have a natural decomposition, we used a depth-first search algorithm to construct a tree decomposition in a greedy fashion for LPQP-T.

BENCHMARKED METHODS We compare the performance of LPQP-U and LPQP-T with the widely used MPE algorithms, TRWS (Kolmogorov 2006) and MPLP (Sontag et al. 2008), both of which are LP relaxations. For both algorithms we used the implementation made available by the authors. These algorithms represent different trade-offs in performance. TRWS is a highly efficient message-passing algorithm for the standard LP relaxation. It is much faster than the MPLP, especially on large instances where the MPLP convergence is pretty slow. MPLP on the other hand, initially solves the LP relaxation over the local polytope, and in later iterations includes additional summation constraints over sets of three or four variables. This strategy naturally leads to lower (better) energy solutions, on instances where the LP relaxation is not tight. The MPLP was shown to identify the global optimum for some problems.

PERFORMANCE MEASURES In this work we mainly compared the quality of the solutions, which in the MPE setting is most naturally measured by the energy associated with an assignment (3.1). Strictly comparing energy values is problematic for two reasons. The values lack proper scaling required for quantitative comparison of different results on the same problem instance, and are not comparable across instances. We therefore exercise the following scoring procedure. Let $e_1, \dots, e_J$ denote the energies of the compared solutions, we set

$$s_i = \frac{\max_{1 \le j \le J}(e_j) - e_i}{\max_{1 \le j \le J}(e_j) - \min_{1 \le j \le J}(e_j)} \tag{3.17}$$

as the score of the i-th method. This scheme assigns the worst and the best methods, scores of zero and one respectively. The remaining methods get a fraction relative to their value between the best and

the worst result. This procedure is not flawless since the scores are still computed relative to the worst energies. It was most often the case though, that TRWS was the lowest scoring method. Being an often used algorithm, using it as a normalizing measure is in our opinion a sensible choice. In experiments where the optimal value is known, we use this value instead of $\min_{1\leq j\leq J} e_j$. In addition to comparing the quality of the solution, we comment about the trends in the efficiency (run-time) of the various methods.

3.6.1 *Synthetic Potts Model Data*

M (size)	60		90		120	
K (# states)	2	5	2	5	2	5
	$\sigma = 0.05$					
MPLP	0.71	0.99	0.51	0.96	0	0.95
LPQP-U	0.97	0.99	0.97	1	0.98	1
LPQP-T	1	0.97	1	0.98	1	0.98
TRWS	0	0	0	0	0.39	0
	$\sigma = 0.5$					
MPLP	1	1	1	1	1	0.99
LPQP-U	0.99	0.92	0.99	0.91	1	0.94
LPQP-T	0.99	0.95	0.99	0.94	0.99	0.96
TRWS	0	0	0	0	0	0

Table 3.1: Averaged scores achieved by the MPE solvers on the synthetic grid data. The scores, computed according to (3.17), assign in each run 1 and 0 to the best and the worst objective values. The remaining algorithms get a fractional score reflecting their relative objective value.

We follow a similar experimental setup as in (Ravikumar, Agarwal, and Wainwright 2010). The graph is a 4-nearest neighbor grid of varying size. We used $M = 60, 90, 120$ where M is the grid side-length, and M^2 is the overall number of variables. We used $K = 2$ and $K = 5$ for the number of states. The unary potentials were randomly set to $\theta_{i;k}(y_i) \sim \mathit{Uniform}(-\sigma, \sigma)$, and for σ we used values in $[0.05, 0.5]$. Note that the problem instance gets harder for small values of σ, this parameter can be understood as the signal-to-noise ratio. The pairwise

potentials $\theta_{ij}(y_i, y_j)$, were set to penalize agreements or disagreements of the labels, by an amount $\alpha_{ij} \sim \mathit{Uniform}(-1, 1)$, chosen at random. We set $\theta_{ij}(y_i, y_j) = 0$ if $y_i \neq y_j$ and α_{ij} otherwise. In this experiment we choose the graph decomposition for the LPQP-T solution as the vertical and horizontal split of the grid edges (see Figure 3.2, left). The two trees have all the original nodes in common, but no overlapping edges.

The results of the comparison using the performance measure given in (3.17), are presented in Table 3.1. For each choice of parameters, we averaged the scores of 5 runs. Furthermore, Figure 3.5 shows the progress of the objective during a run of the LPQP-U algorithm.

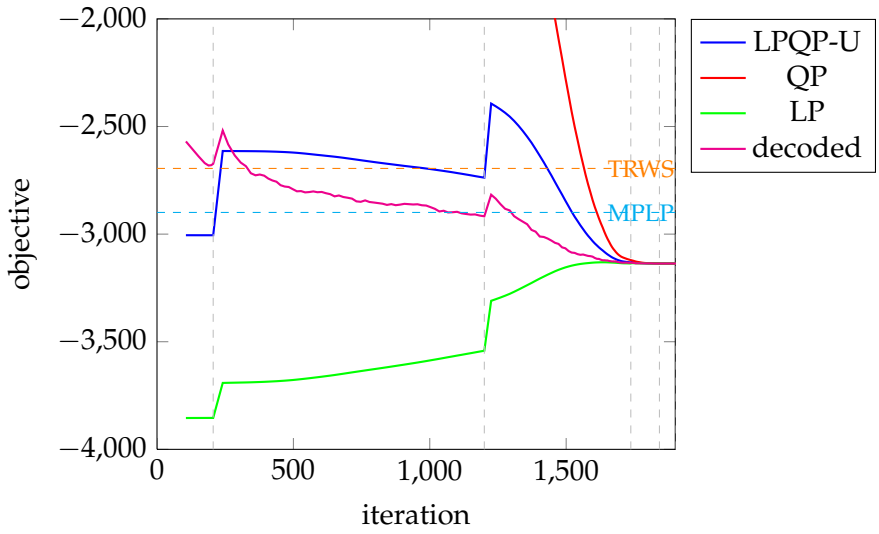


Figure 3.5: Development of the different objectives (for the same $\boldsymbol{\mu}$) during a run of LPQP-U. The decoded objective refers to the current solution independently rounded to integer values. The vertical lines show iterations where ρ was increased. The horizontal lines show the energy of the solution found by TRWS and MPLP, respectively. In the end the auxiliary pairwise variables are consistent with the unary variables, and hence the QP, the LPQP-U and LP objectives all coincide.

In terms of running time, TRWS was always first to output a solution, followed by the LPQP algorithms. MPLP was always slower and on the larger instances did not converge within a predefined maximal time. We therefore restricted the number of tightening iterations of

MPLP to a maximum of 1000. A tightening iteration includes additional constraints into the local marginal polytope. Even after this change, MPLP was still considerably slower than the other algorithms. Between the LPQP algorithms, the LPQP-U was most often faster than LPQP-T.

As we expect, TRWS returned the worst assignment on almost all configurations. The energies obtained by LPQP-U, LPQP-T and MPLP were in general very close. We observe that both of the LPQP algorithms, returned slightly better solutions in comparison to the MPLP, when the potentials were sampled with lower signal-to-noise ratio σ.

The run time of LPQP-T seems to be mostly influenced by the structure of the decomposition. In later experiments where the decomposition consisted of a larger number of trees with more variables in common, the LPQP-T was significantly slower compared to the LPQP-U. In terms of the energy of the solutions, the two algorithms were very similar. For this reason we report from now on the LPQP-U only. The LPQP-T can still be beneficial in settings where the computations are performed on a distributed system.

Figure 3.6 shows the energy of the solution as well as the run time of the LPQP algorithms for different initial ρ_0 for a grid graph of size 40×40 with $K = 3$. We can see that for smaller values of ρ_0 one generally obtains better solutions. For larger values of ρ_0, the optimization problem becomes more and more similar to the standard QP relaxation, as violations in the pairwise marginals are strongly penalized. The run time of the algorithms however also increases substantially for smaller ρ_0, especially for LPQP-T.

3.6.2 *Protein Design and Side-chain Prediction*

The protein inference problem discussed in (Yanover, Meltzer, and Weiss 2006), consists of two tasks: protein side-chain prediction and protein design. For the protein prediction task, it was shown in (Yanover, Meltzer, and Weiss 2006) that only for 30 out of the 370 protein prediction instances, the LP relaxation is not tight. For 28 of them, the true MPE was computed using general integer programming techniques. Figure 3.7 visualizes the results of LPQP and TRWS on these instances. LPQP found the global minimum of roughly 2/3 of these more difficult instances. On the remaining 340 instances, the LP is tight. The LPQP found the global optimum in all but three cases (results are not shown).

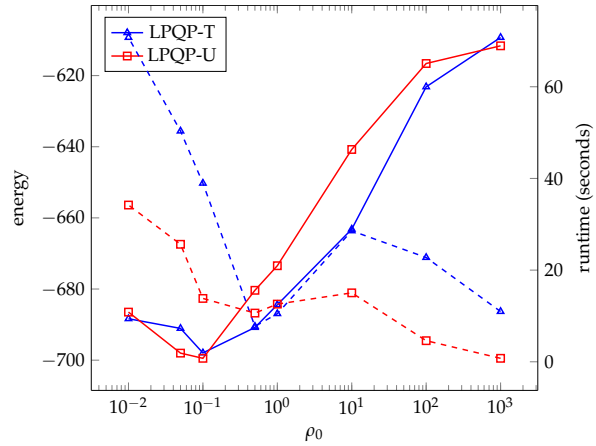


Figure 3.6: Run time (dashed line) and energy of the solution (solid line) found by the LPQP-U and LPQP-T algorithm as a function of the initial ρ_0. For smaller ρ_0 the run time of LPQP-T is much worse affected than the one of LPQP-U. TRWS achieves a solution with an energy of -632.

MPLP was applied to this task in (Sontag et al. 2008), and achieved the global optimum on all instances.

The protein design task consists of 97 instances. We used MPLP to compute the global optimum, but for one of the instances, MPLP did not finish within a time-budget of 7 days. The average scores for the remaining 96 instances are as follows. LPQP-U: 0.93, MPLP: 1 and TRWS: 0.03. The average energies are: LPQP-U: -184.06, MPLP: -184.60, TRWS: -173.55. The QP message-passing algorithm in (Kumar and Zilberstein 2011), was tested on this task as well. The evaluation criteria used in this work was the average (across the 97 instances) percentage of the optimal value. While the reported average value in (Kumar and Zilberstein 2011) is 97.7%, our solution achieves 99.7% percentage of the optimal value on average.

3.6.3 *Decision Tree Fields*

As a last experiment we apply our LPQP algorithm to the recently published "hard discrete energy minimization instances" dataset (Nowozin et al. 2011), available on the authors website. The task is to fill in, or inpaint, a blanked out area in a binary image of Chinese handwritten

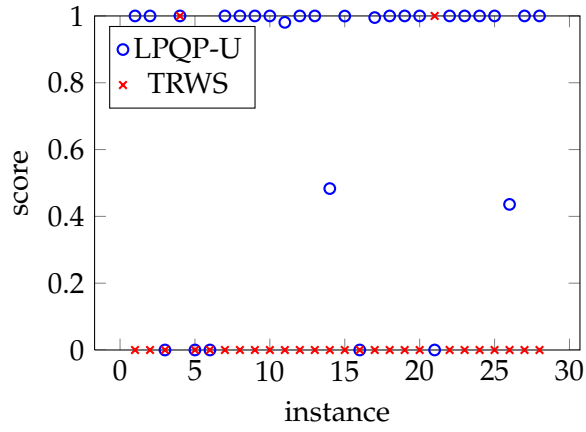


Figure 3.7: Protein prediction results for instances where the LP is not tight. LPQP-U improves on TRWS in all but one cases. For 20 of the 28 instances LPQP-U finds the true MPE.

characters, see Figure 3.8. The dataset consists of 100 energy minimization instances, and comes with approximate MPE solutions obtained using Simulated Annealing (SA) inference, which was found to work better than TRWS. For 43 instances the LPQP algorithm obtained better solutions than the previously best known solutions. Figure 3.8 visualizes some of the instances where the LPQP algorithm leads to a better solution. We observed that the SA solutions seem to hallucinate too much regularity which is not supported by the underlying energy. The scoring of the three algorithms is as follows. LPQP-U: 0.84, SA: 0.74 and TRWS: 0.21. We failed to apply MPLP as the tightening operation did not succeed and the program segfaulted.

3.7 CONCLUSIONS

This chapter introduced a novel formulation for MPE inference in graphical models. The approach combines the LP and QP relaxation terms through a KL divergence measure. The resulting problem, albeit being non-convex, gives rise to efficient algorithms built upon known LP solvers. In almost all experiments we found our LPQP solvers to find better solutions than TRWS. Further, the algorithm is competitive with MPLP in terms of solution, but often more efficient.

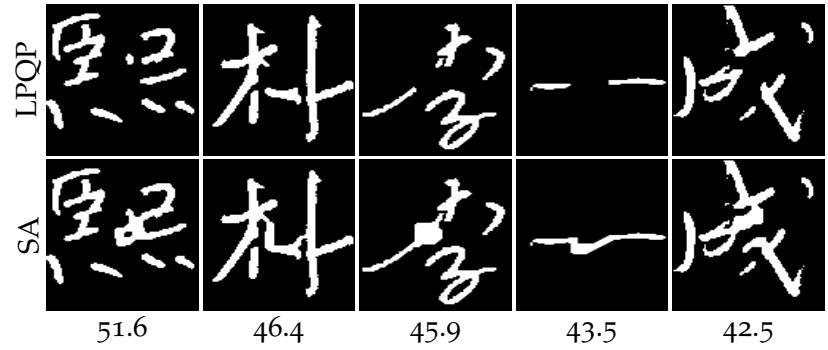


Figure 3.8: Results for the Chinese character inpainting dataset. *Top*: results obtained by LPQP-U. *Middle*: solutions from (Nowozin et al. 2011) obtained by simulated annealing. *Bottom*: Energy difference between the simulated annealing solution and the LPQP solution, the larger the value is, the better the LPQP solution is.

As the LPQP approach can also be understood as a way to perform rounding of an LP solution, it would also be possible to combine LPQP with MPLP to round the solution, in case MPLP can not find any additional constraints to tighten the outer bound.

Another promising direction for future work is to investigate whether one could select the tree decompositions in LPQP-T depending on the value of the penalty function on the edges.

4

LEARNING WITH HIGH-ORDER LOSSES

In all the applications of structured models studied so far we have only considered low-order models, e.g. models with only *pairwise dependencies*. The inability of pairwise models to capture high-order dependencies between random variables restricts their expressive power, and renders them poorly suitable to represent the data well (Sudderth and Jordan 2008). Models containing higher-order factors, on the other hand, are able to encode complex dependencies between groups of variables, and can hence encourage solutions which match the statistics of the ground truth solution better (Potetz 2007; Roth and Black 2009; Woodford, Rother, and Kolmogorov 2009). However, the high computational cost of performing MPE inference in such models has inhibited their use (Lan et al. 2006). Instead, there has been a widespread adoption of the simpler and less powerful pairwise CRF models which allow efficient inference (Szeliski et al. 2008).

This chapter considers a middle ground where in training one is willing to invest in more expensive high-order computations, but at test time only simple and efficient MPE decoding for pairwise models is required. We introduce an exact learning algorithm that estimates the parameters of a pairwise CRF model according to a restricted class of *high-order losses*. As the loss term does not enter the MPE prediction, at test time the prediction task reduces to a simple pairwise energy minimization problem.

More specifically, we consider the margin rescaled maximum-margin learning setting for a *loss function* $\Delta_{\boldsymbol{y}^\star}(\boldsymbol{y})$. The loss term allows a data scientist to guide the parameter learning to focus on specific errors (see Section 2.1). In an ideal world one would like the loss function to match the subsequent evaluation of the structured classifier. Most previous work on structured output learning has considered simple choices of the loss function, such as the Hamming loss or the squared loss, which lead to tractable learning algorithms (Szummer, Kohli, and Hoiem 2008). However, in real world applications, researchers might prefer more general loss functions which penalize deviations in some higher-order statistics.

As an example, the PASCAL Visual Object Classes (VOC) challenge (Everingham et al. 2010) evaluates the overlap between a predicted bounding box B_p and the ground-truth bounding-box B_{gt}:

$$a_0 = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}.$$

Here $B_p \cap B_{gt}$ denotes the intersection of the two bounding boxes and $B_p \cup B_{gt}$ the union. The VOC challenge considers a detection bounding box B_p to be correct, if a_0 is larger than 0.5. Clearly this information is valuable when the parameters of an object detector are estimated. Let us assume that some coding of a bounding box in terms of an output variable vector $\boldsymbol{y}$ is given. Two natural choices for the loss term $\Delta_{\boldsymbol{y}^\star}(\boldsymbol{y})$ in terms of a_0 are:

$$\Delta^{voc}_{\boldsymbol{y}^\star}(\boldsymbol{y}) = a_0 \qquad \text{or} \qquad \Delta^{voc}_{\boldsymbol{y}^\star}(\boldsymbol{y}) = \begin{cases} 1 & \text{if } a_0 < 0.5 \\ 0 & \text{otherwise.} \end{cases}$$

As another example, in medical image analysis for some diagnostic scenarios a physician might be interested in the area of a segmentation of a tumor that is under investigation. Depending on the number of tumor pixels different treatments can then be initiated. It is important to note, that *rather than estimating the label of every individual pixel in a segmentation, the physician is only interested in the area covered by tumor cells*. When applying a structured classifier to such a setting, this information should be provided by means of a loss function to the learning algorithm. Instead of the Hamming loss, a more suitable loss function would be a function that considers the absolute difference between tumor pixels in the ground-truth segmentation and the tumor pixels in the predicted segmentation. However, as we will see, both the VOC bounding box loss and the area loss do not factorize into individual variable terms such as the Hamming loss. This renders maximum-margin learning intractable for these higher-order losses.

This chapter is organized as follows: We introduce the problem setting and loss-augmented inference in Section 4.1, followed by a discussion of low-order and high-order loss functions in Section 4.2. In Section 4.4 we introduce an efficient algorithm for performing loss-augmented inference for a restricted class of high-order loss functions. It uses the lower-envelope representation of higher-order functions (Kohli and Kumar 2010) to transform them to pairwise functions. We test the

efficacy of our approach on the problem of foreground-background segmentation in Section 4.5. The experimental results show that our method is able to obtain parameters which lead to better results compared to the traditional approach.

4.1 PROBLEM SETTING AND LOSS AUGMENTED INFERENCE

In this work we consider binary image labeling problems for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and output variables $y_i \in \{0, 1\}\ \forall i$. The energy is assumed to have the form

$$\begin{aligned} E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) &= -\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle \\ &= \sum_{i \in \mathcal{V}} \theta_i(y_i, \boldsymbol{x}; \boldsymbol{w}^u) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(y_i, y_j, \boldsymbol{x}; \boldsymbol{w}^p). \end{aligned}$$

We leave the exact form of the unary and pairwise potentials unspecified as this is application dependent. The dependence of the potentials on the parameters is assumed to be linear, the standard assumption throughout this thesis. Furthermore, we assume that the pairwise potentials are submodular (or equivalently associative). We restrict ourselves to MPE prediction functions of the form

$$\boldsymbol{y}^\star = \operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{Y}} E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}). \tag{4.1}$$

Due to the binary state space of the individual output variables and the submodularity assumption, exact MPE prediction as in (4.1) is efficiently tractable using the graph-cut algorithm, see Subsection 2.7.6.

Our goal here is to learn the parameters $\boldsymbol{w}$ using the margin-rescaled maximum-margin framework for a high-order loss function. We will make the definition of a high-order loss more precise in Section 4.2. For now one can think of a high-order loss as a complex function comparing a prediction and the ground-truth labeling. The efficient solution of the *loss augmented inference* problem poses the main obstacle to applying maximum margin learning to a structured problem; as most popular solvers for maximum-margin learning are based on a repetitive evaluation of the loss augmented inference oracle. The cutting-plane algorithm discussed in Subsection 2.5.2, stochastic subgradient descent or the Frank-Wolfe algorithms which we recently introduced in (Lacoste-Julien et al. 2013) are examples of such maximum-margin solvers that require an oracle for loss augmented inference. In our work we will

therefore focus on the solution of the loss augmented inference problem, immediately giving rise to several possible ways on how to solve the overall maximum-margin learning problem. In the experiments we will use the cutting-plane algorithm, but the aforementioned solvers could be used instead. Formally, the loss augmented inference problem is given by

$$\min_{\boldsymbol{y} \in \mathcal{Y}} E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) - \Delta_{\boldsymbol{y}^\star}(\boldsymbol{y}). \tag{4.2}$$

The energy minimization problem in (4.2) differs from the one in (4.1) only in that *the negative loss term* is added to the energy. Depending on the form of the loss term, this can render the inference problem intractable. The loss augmented inference problem is investigated in detail in Section 4.4. The next section discusses loss functions in general and introduces the label-count loss, which is promoted in our work.

4.2 LOW-ORDER AND HIGH-ORDER LOSSES

Maximum-margin learning leaves the choice of the loss function $\Delta_{\boldsymbol{y}^\star}(\boldsymbol{y})$ unspecified. The loss allows the researcher to adjust the parameter estimation to the evaluation which follows the learning step. In our work we differentiate between *low-order losses*, which factorize, and *high-order losses*, which do not factorize. Factorization is considered to be a key property of a loss to maintain computational tractability of the loss augmented inference.

4.2.1 *Low-Order Loss Functions*

For image labeling in computer vision a popular choice is the pixelwise error, or also Hamming error (see Section 2.1). It is defined as:

$$\Delta_{\boldsymbol{y}^\star}^{\mathit{Hamming}}(\boldsymbol{y}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} y_i \neq y_i^\star.$$

For image labeling problems, it tries to prevent solutions with high pixel labeling error from having low energy under the model compared to the ground truth.

4.2.2 *High-Order Loss Functions*

In many machine learning applications, practitioners are concerned with errors other than the simple Hamming loss. This is especially the case in medical imaging tasks involving segmentations of particular tissues or tumors. In such problems, radiologists and physicians are sometimes more interested in measuring the exact volume or area of the tumor (or tissue) to analyze if it is increasing or decreasing in size. This preference can be handled during the learning process by using a label-count based loss function.

More formally, consider a two-label image segmentation problem where we have to assign the label 0 (representing 'tumor') or 1 (representing 'non-tumor') to every pixel/voxel in the image/volume. The area/volume based *label-count loss* function in this case is defined as:

$$\Delta_{\boldsymbol{y}^*}^{Count}(\boldsymbol{y}) = \frac{1}{|\mathcal{V}|}\left|\sum_{i\in\mathcal{V}} y_i - \sum_{i\in\mathcal{V}} y_i^\star\right|. \tag{4.3}$$

Such a loss function prevents image labelings (segmentations) with substantially different area/volume compared to the ground truth to be assigned a low energy under the model. As we will show, despite the high-order form of the label-count loss, learning with it in the maximum-margin framework is tractable.

It is easy to show that the label-count loss is a lower bound on the Hamming loss:

$$\Delta_{\boldsymbol{y}^*}^{Count}(\boldsymbol{y}) \leq \Delta_{\boldsymbol{y}^*}^{Hamming}(\boldsymbol{y}).$$

We would like to point out that in case a segmentation is desired, the label-count loss will most likely not lead to accurate results, as it does not relate the location of the foreground pixels to the ones in the ground-truth. In our experiments we however did not observe severe degradations of the segmentation accuracy. This can potentially be attributed to the low-dimensional weight vector and relatively strong unary features. In case an accurate segmentation and label-count is needed, one could also combine the two losses. One advantage of the label-count loss is that it requires much weaker supervision, as only the number of foreground pixels is required, rather than a pixel-accurate segmentation.

4.3 RELATED WORK

In (Lempitsky and Zisserman 2010) a learning approach for counting is introduced. The major difference to our work stems from the model that is learned. In their work a continuous regression function is trained, which predicts for each pixel a positive scalar real value independent of all its neighboring pixels. In our work a CRF is used, which includes dependencies among variables, only the loss term in learning is changed. (Gould 2011) discusses maximum-margin parameter learning in graphical models that contain potentials with a linear lower envelope representation. However, the loss function used in their work is still restricted to be a simple Hamming loss. The idea of learning with higher-order losses is also studied in (Tarlow and Zemel 2011; Tarlow and Zemel 2012) and (Ranjbar et al. 2012). They discuss several higher-order loss functions, but only approximate algorithms are presented. To the best of our knowledge, our work introduces for the first time a subclass of high-order loss functions, for which structured maximum margin learning remains tractable.

4.4 LOWER ENVELOPES REPRESENTATION

Even on its own, the problem of minimizing a general energy function of discrete variables is a NP-hard problem. In this chapter, due to the assumptions of submodularity and binary variables, the energy minimization problem $\min_{\boldsymbol{y}} E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})$ can be solved exactly. The presence of the loss term in the loss augmented energy minimization problem in (4.2) has the potential to make it harder to minimize. The Hamming loss, however, has the nice property that it decomposes into unary terms which can be integrated into the unary energy terms. Therefore the loss augmented inference problem in this case is not harder than standard inference:

$$\min_{\boldsymbol{y}} E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) - \Delta_{\boldsymbol{y}^\star}^{Hamming}(\boldsymbol{y}) =$$
$$\min_{\boldsymbol{y}} \sum_{i \in \mathcal{V}} \underbrace{\left[\theta_i(y_i, \boldsymbol{x}; \boldsymbol{w}^u) - \frac{1}{|\mathcal{V}|} \mathbb{I}_{[y_i \neq y_i^\star]}(y_i) \right]}_{:= \tilde{\theta}_i(y_i)} + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(y_i, y_j, \boldsymbol{x}; \boldsymbol{w}^p).$$

We attribute the popularity of the Hamming loss in image labeling at least partially to the fact that loss augmented inference is no harder

than the initial model energy minimization. In many applications there would exist more meaningful loss functions, that however complicate the loss augmented inference and are therefore not used in practice.

4.4.1 *Compact Representation of High-Order Losses*

While it is easy to incorporate the Hamming loss in the learning formulation, this is not true for higher-order loss functions. In fact, a general M order loss function defined on K-state variables can require up to K^M parameters for just its definition. In recent years a lot of research has been channeled towards developing compact representations of higher-order functions (Kohli, Kumar, and Torr 2009; Rother et al. 2009; Kohli and Kumar 2010). In particular, Kohli and Kumar (2010) proposed a representation based on *upper* and *lower* envelopes of linear functions which enables the use of many popular classes of higher-order potentials employed in computer vision. More formally, they represent higher-order functions as:

$$f^h(\boldsymbol{y}) = \otimes_{q \in \mathcal{Q}} f^q(\boldsymbol{y})$$

where $\otimes = \{\max, \min\}$, and $\mathcal{Q}$ indexes a set of linear functions, defined as

$$f^q(\boldsymbol{y}) = \mu^q + \sum_{i \in \mathcal{V}} \sum_{a \in \mathcal{L}} \nu_{ia}^q \mathbb{I}_a(y_i)$$

where the weights ν_{ia}^q and the constant term μ^q are the parameters of the *linear* function $f^q(\cdot)$, and the function $\mathbb{I}_a(y_i)$ returns 1 if variable y_i takes label a and returns 0 for all other labels. Finally, $\mathcal{L}$ denotes the set of possible states for an output variable, for example in binary image denoising $\mathcal{L} = \{0, 1\}$. While $\otimes = \min$ results in a lower envelope of the linear functions, $\otimes = \max$ results in the upper envelope.

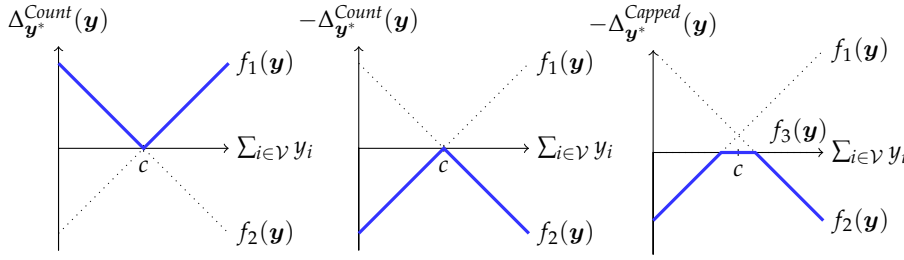
The upper envelope representation, in particular, is very powerful and is able to encode sophisticated silhouette constraints for 3D reconstruction (Kohli and Kumar 2010; Kolev and Cremers 2008). It can also be used to compactly represent general higher-order energy terms which encourage solutions to have a particular distribution of labels. Woodford, Rother, and Kolmogorov (2009) had earlier shown that such terms were very useful in formulations of image labeling problems such as image denoising and texture synthesis, and led to better results.

The higher-order label-count loss defined in (4.3) can be represented by taking the upper envelope of two linear functions $f^1(\cdot)$ and $f^2(\cdot)$ that are defined as:

$$f^1(\boldsymbol{y}) = \frac{1}{|\mathcal{V}|}\left(\sum_{i\in\mathcal{V}} y_i - \sum_{i\in\mathcal{V}} y_i^\star\right), \tag{4.4}$$

$$f^2(\boldsymbol{y}) = \frac{1}{|\mathcal{V}|}\left(\sum_{i\in\mathcal{V}} y_i^* - \sum_{i\in\mathcal{V}} y_i\right). \tag{4.5}$$

This is illustrated in Figure 4.1a.



(a) Upper envelope. (b) Lower envelope. (c) Capped loss.

Figure 4.1: Upper and lower envelope representations of the label-count loss and its negation. Here $c := \sum_{i\in\mathcal{V}} y_i^*$. Interestingly, as the loss enters the loss-augmented energy with a negative sign, the resulting energy minimization problem $\min_{\boldsymbol{y}} E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) - \Delta_{\boldsymbol{y}^*}^{count}(\boldsymbol{y})$ becomes tractable. (c) shows an example of a loss which can be described as the lower envelope of three linear functions. For visualization purposes we dropped the scaling by $1/|\mathcal{V}|$ in front of the loss.

4.4.2 *Minimizing Loss Augmented Energy Functions*

Although upper envelope functions are able to represent a large class of useful higher order functions, inference in models containing upper envelope potentials involves the solution of a hard min-max optimization problem (Kohli and Kumar 2010). Interestingly, the loss term in the loss-augmented energy minimization problem (4.2) has a *negative* coefficient, which allows us to represent the label-count loss (4.3) by

the *lower envelope* of the functions defined in (4.4) and (4.5) (visualized in Figure 4.1b).

Kohli and Kumar showed that the minimization of higher order functions that can be represented as lower envelopes of linear functions can be transformed to the minimization of a pairwise energy function with the addition of an auxiliary variable. In fact, in some cases, the resulting pairwise energy function can be shown to be submodular (Boros and Hammer 2002; Kolmogorov and Zabih 2004) and hence can be minimized by solving a minimum cost s-t-cut problem (Kohli, Ladicky, and Torr 2009). This is the case for all higher-order functions of Boolean variables which are defined as:

$$f^h(\boldsymbol{y}) = \mathcal{F}\left(\sum_{i \in \mathcal{V}} y_i\right),$$

where $\mathcal{F}$ is a concave function. The worst case time complexity of the procedure described above is polynomial in the number of variables. A related family of higher-order submodular functions which can be efficiently minimized was characterized in (Stobbe and Krause 2010). Next, we consider the loss augmented inference for the label-count loss in more detail.

4.4.3 *Label-Count Loss Augmented Inference*

The minimization of the negative label-count loss (4.3) can be transformed to the following pairwise submodular function minimization problem (ignoring the scaling by $1/|\mathcal{V}|$):

$$\begin{aligned}
\min_{\boldsymbol{y}} -\Delta^{Count}_{\boldsymbol{y}^*}(\boldsymbol{y}) &= \min_{\boldsymbol{y}} -\left|\sum_{i \in \mathcal{V}} y_i - \sum_{i \in \mathcal{V}} y_i^\star\right| \\
&= \min_{\boldsymbol{y}, z \in \{0,1\}} -z\left(\sum_{i \in \mathcal{V}} y_i - \sum_{i \in \mathcal{V}} y_i^\star\right) \\
&\qquad -(1-z)\left(\sum_{i \in \mathcal{V}} y_i^\star - \sum_{i \in \mathcal{V}} y_i\right) \\
&= \min_{\boldsymbol{y}, z \in \{0,1\}} 2z\left(\sum_{i \in \mathcal{V}} y_i^\star - \sum_{i \in \mathcal{V}} y_i\right) + \sum_{i \in \mathcal{V}} y_i - \sum_{i \in \mathcal{V}} y_i^\star.
\end{aligned} \tag{4.6}$$

The full energy minimization for the label-count loss augmented inference reads as follows

$$\min_{\boldsymbol{y},z\in\{0,1\}} E(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w}) + \frac{1}{|\mathcal{V}|}\left(2z\left(\sum_{i\in\mathcal{V}} y_i^\star - \sum_{i\in\mathcal{V}} y_i\right) + \sum_{i\in\mathcal{V}} y_i - \sum_{i\in\mathcal{V}} y_i^\star\right) \quad (4.7)$$

As we assume that the original energy $E(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w})$ is submodular, the pairwise problem above is exactly solved by graph-cut (Boykov 2001). Next, we illustrate the construction of the pairwise graphical model, which can then be solved using graph-cut. We add one node for the variable z to the original graph. This auxiliary node is then connected to every segmentation variable y_i, adding a total of $|\mathcal{V}|$ new edges to the graph. The pairwise energy construction is visualized in Figure 4.2. Unfortunately, we found the de-facto standard computer vision graph-

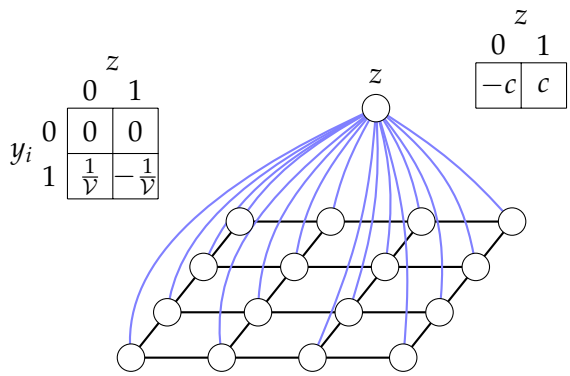


Figure 4.2: Pairwise graph used for solving the label-count loss augmented inference problem. The potentials of the edges connecting the segmentation nodes y_i to the auxiliary node z (which are shown in blue) are visualized to the left. The unary potential of the auxiliary variable z to the right, where $c := \frac{1}{|\mathcal{V}|}\sum_i y_i^*$. Standard graph-cut solvers can be applied to this problem.

cut algorithm by Boykov and Kolmogorov (2004) to run fairly slowly on these problem instances. We attribute this to the dense connectivity of the auxiliary node z. This problem is in theory, and as is turns out also in practice, solved by the recent Incremental Breadth First Search (IBFS) graph-cut algorithm introduced in (Goldberg et al. 2011). We found this algorithm to be roughly an order of magnitude more efficient than the Boykov-Kolmogorov algorithm. Learning on a small subset of the data

discussed in the next section took two minutes when IBFS was used and around 25 minutes with the Boykov-Kolmogorov algorithm.

Alternatively, for minimizing the loss augmented energy with a single Boolean z, as in (4.7), one can solve the minimization efficiently by performing energy minimization twice in the original graph (for $z = 0$ and $z = 1$). Each choice of z results in different unaries. This approach does however not scale to the case where the loss augmented energy has multiple variables z, as the number of sub-problems grows exponentially. If we have a loss function with 10 zs one will have to perform the minimization 2^{10} times. The case of several variables z could potentially be interesting if one wants to penalize the label-count loss within several bounding boxes in an image.

4.5 EXPERIMENTS

We implemented the maximum margin learning in Matlab. For solving the QP the MOSEK solver is used. Loss augmented inference is performed by IBFS, which is implemented in C++ through a MEX wrapper. The IBFS code was downloaded from the authors website and modified to support double precision energies (as opposed to integer precision). Submodularity of the model is explicitly enforced in training by ensuring that all the edge potentials off-diagonal energies are larger than the diagonal energies. This can be achieved by adding additional constraints to the QP. We do so by the ensuring that the non-diagonal pairwise weight is positive. Combined with the fact that all the edge features by construction are positive and the diagonal weights are set to zero, this ensures submodularity. In all of the experiments the regularization parameter λ is chosen on a validation set. The final results are then generated from a held out test set. The running time of the label-count loss based learning is roughly one order of magnitude slower than the standard Hamming loss learning. The trick discussed earlier of simply performing graph-cut inference twice for different unaries should get the runtime increase down to a factor of around two.

4.5.1 *Mitochondria Cell Segmentation*

Counting tasks naturally arise in many medical applications. The estimation of the progression of cancer in a tissue or the density of cells

in microscope images are two examples. As a first experiment we study the problem of counting the number of mitochondria cell pixels in an image. The dataset is visualized in Figure 4.3. The images have been provided by Ángel Merchán and Javier de Felipe from the Cajal Blue Brain team at the Universidad Politécnica de Madrid. Three images

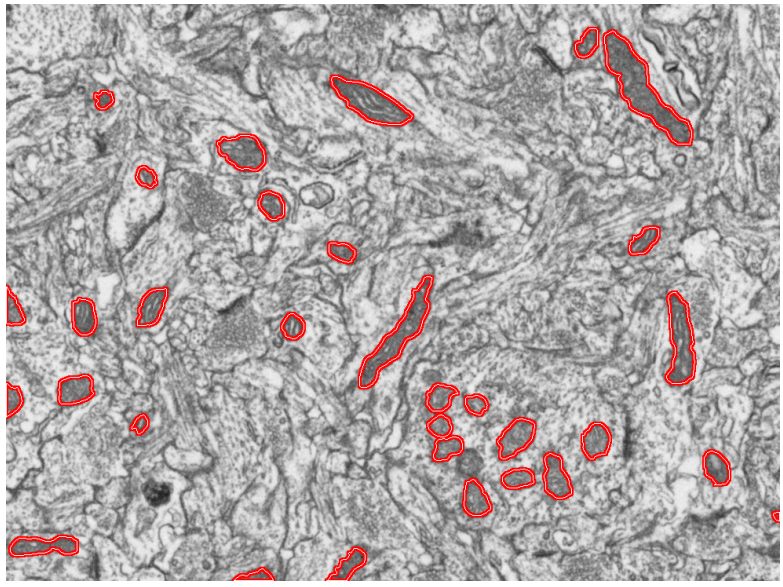
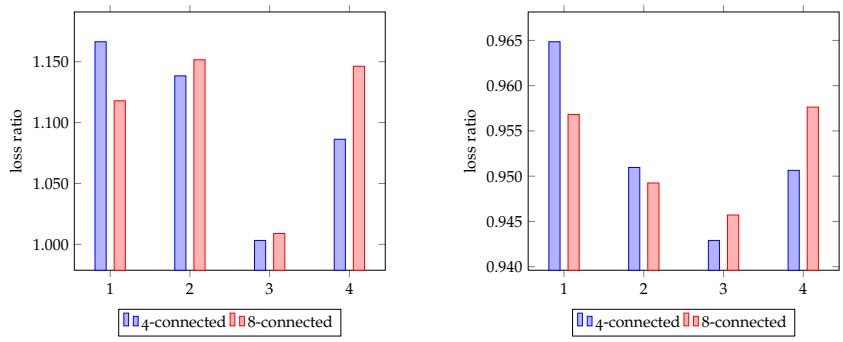


Figure 4.3: Electron microscopy image showing the mitochondria cells in red.

are used for learning, two images for the validation and the remaining five images for testing. The images have a resolution of 986×735. The pairwise CRF consists of a unary term with three features (the response of a unary classifier for mitochondria and synapse detection and an additional bias feature). The pairwise term incorporates two features (the color difference between neighboring pixels and a bias). The results for four different random data splits are shown in Figure 4.4. As expected the label-count loss trained model performs better than the Hamming loss trained model if the label-count loss is used for the evaluation and vice-versa if evaluated on the Hamming loss. The same trend is observed if the 8-connected grid graph is used instead of the standard 4-connected grid graph. We also compared our lower envelope inference approach to the COMPOSE max-product algorithm (Duchi et al. 2006) which is used in (Tarlow and Zemel 2011; Tarlow and Zemel



(a) Label-count loss evaluation. (b) Hamming loss evaluation.

Figure 4.4: Test set results for the mitochondria segmentation on four random data splits. We plot the loss ratio of the Hamming loss trained model and the label-count loss trained model. If the label-count trained model performs better, the ratio is larger than one and smaller than one, in case the Hamming loss trained model is better. (a) Shows the results if the label-count loss is used for the evaluation and (b) shows the results if instead the Hamming loss is used for the evaluation. As one would expect, the model that matches the evaluation loss performs better, which shows that our learning approach works well.

2012). Max-product inference is in general only approximate. However, for the cell segmentation problem in combination with the label-count loss, the solutions obtained using the two different loss augmented inference algorithms were almost identical. The running time of the two approaches is also comparable. Our inference algorithm is slightly more efficient, but also more adapted to the count-loss.

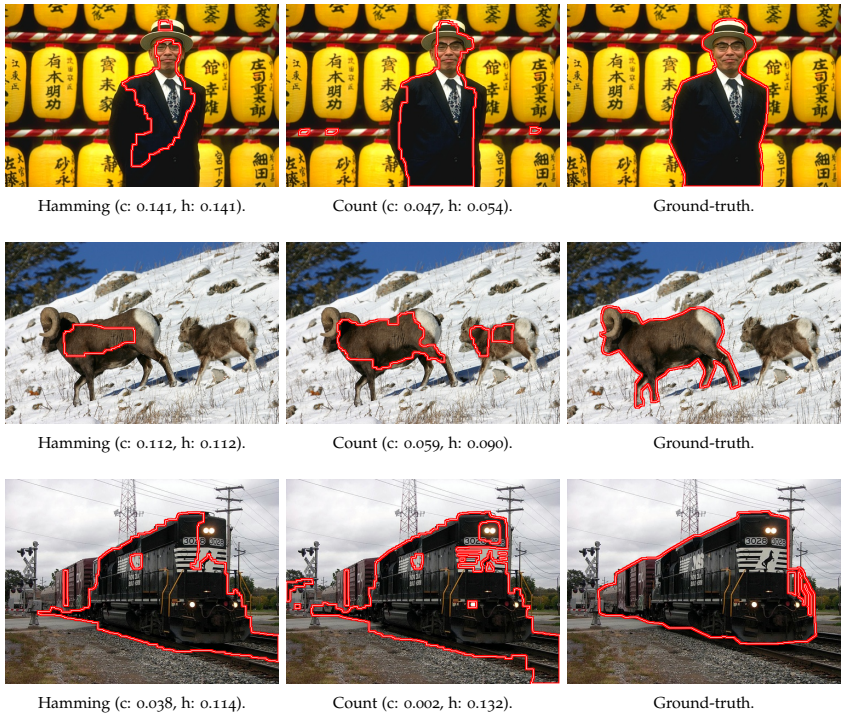
4.5.2 *Foreground-Background Segmentation*

We check the effectiveness of the label-count loss foreground-background segmentation on the Grabcut dataset (Blake et al. 2004). We use the extended dataset from (Gulshan et al. 2010). The dataset consists of 151 images, each comes with a ground truth segmentation. Furthermore, for each image an initial user seed is specified by strokes marking pixels belonging to the foreground or to the background, respectively.

As unary features we use the three color channels together with the background and foreground posterior probabilities as computed by the Gaussian mixture model algorithm used in Grabcut. Additionally we also include a constant feature to correct for class bias. For the pairwise features we use the color difference between the two pixels and again a bias feature. The standard four-connected grid graph is used as the basic model. Each edge is parametrized by the same parameter. We also experimented with extensions of this basic model: In one variant we consider the eight-connected grid, in the other variant each direction of the edge is parametrized using a different parameter. The basic model is therefore specified by an eight dimensional weight vector $\boldsymbol{w}$, the eight-connected model where each direction has its own parameter by a 14-dimensional $\boldsymbol{w}$. For learning 60 images were used, 20 for the validation of the regularization parameter λ, the remaining 71 images were used for testing. Figure 4.5 shows some of the learned segmentations and Figure 4.6 gives a comparison of the models trained using the Hamming loss and the label-count loss. The results in Figure 4.6 show the loss ratio for four different data splits. As expected, we observe that if the label-count loss is used for the evaluation, the model that is trained using this loss performs superior and vice-versa for the Hamming loss. The improvement is consistent across different model structures as can be seen in Figure 4.6.

4.6 DISCUSSION

We have demonstrated, for the first time, how low-order models like pairwise CRFs can be encouraged to preserve higher-order statistics by introducing high-order loss functions in the learning process. The learning involves the minimization of the loss augmented energy, which we show can be performed exactly for certain loss functions by employing a transformation scheme. We demonstrate the efficacy of our method by using a label-count loss while learning a pairwise CRF model for binary image segmentation. The label-count loss function is useful for applications that require the count of positively labeled pixels in an image to match the count observed on a ground truth segmentation. Our proposed algorithm enables efficient maximum margin learning under the label-count loss, and leads to models that produce solutions with statistics that are closer to the ground-truth, compared to solutions of models learned using the standard Hamming loss.

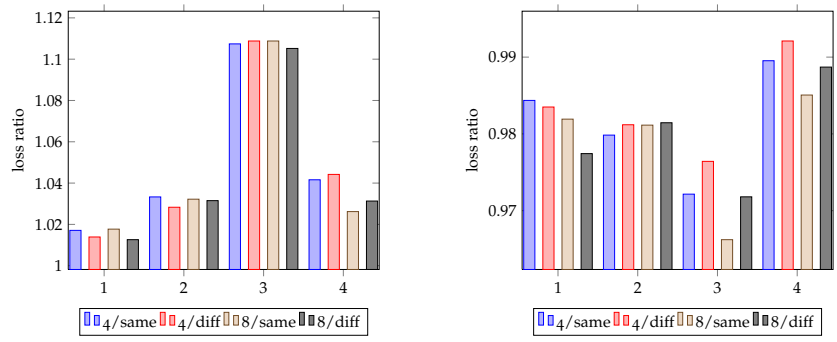


Hamming (c: 0.141, h: 0.141). Count (c: 0.047, h: 0.054). Ground-truth.

Hamming (c: 0.112, h: 0.112). Count (c: 0.059, h: 0.090). Ground-truth.

Hamming (c: 0.038, h: 0.114). Count (c: 0.002, h: 0.132). Ground-truth.

Figure 4.5: Segmentations on the test set for models trained using the Hamming loss (left) and the label-count loss (middle). The image on the right shows the ground-truth segmentation. We show the measured label-count loss and Hamming loss in brackets. The bottom row shows a case where the model trained using the label-count loss shows a better count loss, however the Hamming loss deteriorates due to the false positives. For the first two images, the label-count loss trained model even outperforms the Hamming loss trained model in terms of Hamming loss.



(a) Label-count loss evaluation.

(b) Hamming loss evaluation.

Figure 4.6: Test set performance for the Grabcut dataset on four random data splits. As in the mitochondria segmentation application, we illustrate the loss ratio of the Hamming loss trained model and the label-count trained model. A ratio larger than one implies that the label-count trained model is better, a ratio smaller than one that the Hamming trained model performs better. (a) Shows the results if the label-count loss is used for the evaluation and (b) shows the results if instead the Hamming loss is used for the evaluation. As for the mitochondria segmentation task, the experiments are consistent with the theory in the sense that the model that matches the evaluation loss performs better. The different bars in the illustration show different model structures (4 vs. 8 grid and the same vs. different parametrization of the edges).

5 MAXIMUM ENTROPY AND MAXIMUM MARGIN

This chapter introduces a unified surrogate loss which generalizes the CRF and the structured SVM. This framework allows us to analyze and contrast the two different structured output learning approaches. A continuous parameter interpolates between the two extreme cases and gives rise to a family of learning approaches. Moreover, the surrogate loss is also applicable to latent variable models and direct loss minimization, which we will demonstrate.

5.1 A UNIFIED LOSS FOR STRUCTURED OUTPUT LEARNING

In this section we derive our generalized surrogate loss. First, the CRF log-loss is modified through incorporating an *inverse temperature* parameter. The concept of a margin is introduced into this modified surrogate loss, resulting in a new family of surrogate loss functions.

As already introduced in Subsection 2.5.1, the CRF considers a log-linear model

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{w})} \exp\Big(\big\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})\big\rangle\Big),$$

with the partition function

$$Z(\boldsymbol{x},\boldsymbol{w}) = \sum_{\boldsymbol{y}\in\mathcal{Y}} \exp\Big(\big\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})\big\rangle\Big).$$

The log-loss can be derived as the negative log-likelihood of the probabilistic conditional model

$$\ell_{ll}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y}) := -\log P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w}) = -\big\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})\big\rangle + \log Z(\boldsymbol{x},\boldsymbol{w}).$$

Using the log-loss in the regularized training objective in (2.17) together with an L_2 regularizer corresponds to maximum-a-posteriori (MAP) parameter estimation, where we assume a Gaussian prior on $\boldsymbol{w}$.

The maximum margin principle gives rise to an alternative choice for a structured surrogate loss which is employed in the structured

SVM (see Subsection 2.5.2). The ground-truth output is compared to the output that maximizes the inner product

$$\ell_{mm}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) := -\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle + \max_{\boldsymbol{y}' \in \mathcal{Y}} \left[\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}') \rangle + \Delta_{\boldsymbol{y}}(\boldsymbol{y}') \right]. \quad (5.1)$$

Here, $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$ ensures a margin between the ground-truth output $\boldsymbol{y}$ and an output $\boldsymbol{y}'$. $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$ is discussed in more detail in Section 2.1. Sometimes we will find it more convenient to write the max-margin loss using the *difference of feature maps* $\boldsymbol{\psi}(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}') - \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$:

$$\ell_{mm}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) := \max_{\boldsymbol{y}' \in \mathcal{Y}} \left[\langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y}) \rangle + \Delta_{\boldsymbol{y}}(\boldsymbol{y}') \right].$$

5.1.1 *Inverse Temperature*

As it is done in (2.11), we now introduce a scalar parameter into the log-linear model of the CRF which allows us to control the peakedness of the distribution. For the posterior, we consider the Gibbs distribution with an inverse temperature $\beta \in \mathbb{R}^+$:

$$P_\beta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z_\beta(\boldsymbol{x}, \boldsymbol{w})} \exp\Big(\beta \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle\Big), \quad (5.2)$$

with corresponding normalization constant

$$Z_\beta(\boldsymbol{x}, \boldsymbol{w}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\Big(\beta \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle\Big).$$

For $\beta = 1$ this reverts to the standard CRF. The inverse temperature β does not have any influence on the MPE prediction for an input $\boldsymbol{x}$:

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} P_\beta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle.$$

However, the learning objective is affected by the introduction of the inverse temperature. For reasons that will become clear later on, we choose to scale the per-example surrogate loss by $1/\beta$. The negative log-loss for an instance $(\boldsymbol{x}, \boldsymbol{y})$ thus becomes

$$-\frac{1}{\beta} \log P_\beta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = -\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle + \frac{1}{\beta} \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp\Big(\beta \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}') \rangle\Big).$$

Rearranging terms, it can be shown that the introduction of β is equivalent to changing the L_2 regularizer weight λ in a standard CRF objective

to $\lambda' = \lambda/\beta$ (see below). Hence without further modification to the surrogate loss, β is simply redundant. In the next sections we will show that if the inclusion of the inverse temperature is combined with an additional loss-term, interesting connections emerge.

Proof. Our goal here is to show that

$$\operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} -\frac{1}{\beta} \log P_\beta(\boldsymbol{y}^n|\boldsymbol{x}^n, \boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2, \tag{5.3}$$

is equivalent to ERM with the standard log-loss with regularizer parameter $\lambda' = \lambda/\beta$.

$$\begin{aligned}
&\operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} -\frac{1}{\beta} \log P_\beta(\boldsymbol{y}^n|\boldsymbol{x}^n, \boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \\
&\quad\Leftrightarrow \operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} -\log P_\beta(\boldsymbol{y}^n|\boldsymbol{x}^n, \boldsymbol{w}) + \frac{\beta\lambda}{2}\|\boldsymbol{w}\|_2^2 \\
&\quad\Leftrightarrow \operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} -\log P(\boldsymbol{y}^n|\boldsymbol{x}^n, \boldsymbol{w}) + \frac{\beta\lambda}{2}\left\|\frac{\boldsymbol{w}}{\beta}\right\|_2^2 \\
&\quad\Leftrightarrow \operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} -\log P(\boldsymbol{y}^n|\boldsymbol{x}^n, \boldsymbol{w}) + \frac{\lambda}{2\beta}\|\boldsymbol{w}\|_2^2.
\end{aligned}$$

And thus $\lambda' = \lambda/\beta$. □

5.1.2 *Large Margin Learning*

A standard CRF considers unbiased output distributions. Motivated by the concept of large margin learning, we bias the conditional distribution of outputs $\boldsymbol{y}'$, given the ground-truth output $\boldsymbol{y}$, to have a large margin for outputs $\boldsymbol{y}'$ that are dissimilar. To do so, we assume that a non-negative loss term $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$ is given (also see Section 2.1). The loss term $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$ specifies a preference on the outputs $\boldsymbol{y}'$ when compared to the ground-truth output $\boldsymbol{y}$. In the coming subsection we will incorporate the margin principle of structured SVMs into the conditional probabilistic model given in (5.2).

5.1.3 *Combining the Posterior and the Loss*

The training phase exploits two sources of information: $\Delta_{\boldsymbol{y}^\star}(\boldsymbol{y})$ and $P_\beta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$. In principle, there are many choices for combining the two

sources over the same output variable $\boldsymbol{y}$. Here, we specifically discuss two choices corresponding to *slack and margin rescaling* in the structured SVM (Tsochantaridis et al. 2005).

MARGIN RESCALING For a given ground-truth output $\boldsymbol{y}$, the loss $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$ is transformed into conditional probabilities over outputs:

$$P_\beta(\boldsymbol{y}'|\boldsymbol{y}) \propto \exp\big(\beta\Delta_{\boldsymbol{y}}(\boldsymbol{y}')\big). \tag{5.4}$$

For outputs $\boldsymbol{y}'$ which are very different from the ground-truth $\boldsymbol{y}$, $P(\boldsymbol{y}'|\boldsymbol{y})$ is large. In training this information is used to make such outputs to be difficult to separate, forcing the classifier to ensure good classification on these outputs. $P_\beta(\boldsymbol{y}'|\boldsymbol{y})$ is only leveraged on in training, in order to bias the estimated posterior $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w})$ away from poor outputs $\boldsymbol{y}$ according to the loss.

The first option of combining the posterior and error term is by multiplying (5.2) and (5.4).

$$P_\beta(\boldsymbol{y}'|\boldsymbol{y},\boldsymbol{x},\boldsymbol{w}) \propto P_\beta(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{w})P_\beta(\boldsymbol{y}'|\boldsymbol{y})$$

Ensuring normalization of the probability distribution leads to

$$P_\beta(\boldsymbol{y}'|\boldsymbol{y},\boldsymbol{x},\boldsymbol{w}) = \frac{1}{Z_\beta(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w})}\exp\Big(\beta\langle\boldsymbol{w},\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}')\rangle + \beta\Delta_{\boldsymbol{y}}(\boldsymbol{y}')\Big), \tag{5.5}$$

where the partition function is given by

$$Z_\beta(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w}) = \sum_{\boldsymbol{y}'\in\mathcal{Y}}\exp\Big(\beta\langle\boldsymbol{w},\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}')\rangle + \beta\Delta_{\boldsymbol{y}}(\boldsymbol{y}')\Big).$$

Note that the distribution of an output $\boldsymbol{y}'$ is now conditioned on the true output $\boldsymbol{y}$. We do this to ensure good separation of $\boldsymbol{y}$ to outputs $\boldsymbol{y}'$ that are unfavorable according to $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$. In Section 5.2 we show that combining the two posteriors by means of a product, corresponds to *margin rescaling* in the structured SVM case.

For convenience, the loss is absorbed into the feature map by including $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$ as an additional feature: $\boldsymbol{\phi}_\Delta(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{y}) = [\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}')^\mathsf{T}, \Delta_{\boldsymbol{y}}(\boldsymbol{y}')]^\mathsf{T}$. The $\boldsymbol{w}$ needs to be adjusted accordingly by $\boldsymbol{w}_\Delta = [\boldsymbol{w}^\mathsf{T}, 1]^\mathsf{T}$. The score of the ground-truth output $\boldsymbol{y}$ remains unchanged by the introduction of the loss, i.e., $\langle\boldsymbol{w},\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})\rangle = \langle\boldsymbol{w}_\Delta,\boldsymbol{\phi}_\Delta(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{y})\rangle$, as $\Delta_{\boldsymbol{y}}(\boldsymbol{y}) = 0$.

Under this transformation, the surrogate loss of an example $(\boldsymbol{x},\boldsymbol{y})$ is defined as the negative log-likelihood of the conditional probability

in (5.5). As before, rescaling the surrogate loss by $1/\beta$ yields to the *soft-max loss*:

$$\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = -\big\langle \boldsymbol{w}_\Delta, \boldsymbol{\phi}_\Delta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{y})\big\rangle + \frac{1}{\beta}\log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp\Big(\beta\big\langle \boldsymbol{w}_\Delta, \boldsymbol{\phi}_\Delta(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y})\big\rangle\Big). \quad (5.6)$$

We advocate $\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$ as a surrogate loss for structured outputs, generalizing both CRF and structured SVM. Alternatively, one can rewrite the soft-max loss as a single partition function by absorbing the ground-truth contribution into the exponent:

$$\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\beta}\log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp\Big(\beta\big\langle \boldsymbol{w}_\Delta, \boldsymbol{\psi}_\Delta(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y})\big\rangle\Big). \quad (5.7)$$

Here we define $\boldsymbol{\psi}_\Delta(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y}) := [\boldsymbol{\psi}(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y})^\mathsf{T}, \Delta_{\boldsymbol{y}}(\boldsymbol{y}')]^\mathsf{T}$.

SLACK RESCALING An alternative option for combining the conditional probability $P_\beta(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{w})$ with the loss $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$, corresponds to slack rescaling in the structured SVM. The structured SVM with slack rescaling differs in the constraint, but otherwise has the same objective as the margin rescaled structured SVM in (2.22):

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\xi}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{N}\sum_{n=1}^{N} \xi^n \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}^n)\rangle - \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y})\rangle \geq 1 - \frac{\xi^n}{\Delta_{\boldsymbol{y}^n}(\boldsymbol{y})} \quad \forall \boldsymbol{y} \backslash \boldsymbol{y}^n. \end{aligned} \quad (5.8)$$

Note that the ground-truth output $\boldsymbol{y}^n$ is excluded from the constraint set. The slack rescaled structured SVM can also be written as an ERM for the surrogate loss by rewriting the constraint in terms of the slack variable:

$$\ell^{sl}_{mm}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) := \max_{\boldsymbol{y}' \in \mathcal{Y} \backslash \boldsymbol{y}} \left[\Delta_{\boldsymbol{y}}(\boldsymbol{y}')\left(1 + \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y})\rangle\right)\right],$$

One desirable property of slack rescaling, contrary to margin rescaling, is the property that the optimal weight of (5.8) is invariant to scaling of the loss $\Delta_{\boldsymbol{y}^n}(\boldsymbol{y})$. For additional details about the differences between margin and slack rescaling, see (Tsochantaridis et al. 2005, Section 2.2.5).

In our setting, slack rescaling corresponds to choosing the posterior distribution over outputs as follows:

$$P_\beta^{sl}(\boldsymbol{y}'|\boldsymbol{y},\boldsymbol{x},\boldsymbol{w}) = \frac{1}{Z_\beta^{\text{sl}}(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w})} \exp\left(\beta\Big(1 + \big\langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{y})\big\rangle\Big)\right)^{\Delta_{\boldsymbol{y}}(\boldsymbol{y}')},$$

with corresponding partition function $Z_\beta^{\text{sl}}(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w})$. The posterior distribution can be motivated by a soft-max of the scoring function which is used in slack rescaling. The resulting scaled negative log-likelihood that corresponds to the multiplicative factor in *slack rescaling* is given by

$$\ell_\beta^{sl}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y}) = \frac{1}{\beta}\log \sum_{\boldsymbol{y}'\in\mathcal{Y}} \exp\left(\beta\Delta_{\boldsymbol{y}}(\boldsymbol{y}')\Big(1 + \big\langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{y})\big\rangle\Big)\right).$$

We will refer to this surrogate loss as the *soft-max loss with slack rescaling*. Note that in this form there is no ground-truth term in front of the sum over all the outputs $\boldsymbol{y}'$. Again, the error term corresponds to a modification of the feature map. Thus, we arrive at the soft-max given in (5.7) where we use $\boldsymbol{\psi}_\Delta^{sl}(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{y}) = \Delta_{\boldsymbol{y}}(\boldsymbol{y}')[\boldsymbol{\psi}(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{y})^\mathsf{T}, 1]^\mathsf{T}$ instead of $\boldsymbol{\psi}_\Delta(\boldsymbol{y}'|\boldsymbol{x},\boldsymbol{y})$. The reader should notice the non-linear nature of this combination, which makes slack rescaling computationally more challenging than margin rescaling.

The probabilistic interpretation of margin rescaling is more appealing due to the factorization into two posterior distributions. We will therefore concentrate our analysis on margin rescaling. Nevertheless, most of the findings also hold for slack rescaling.

5.2 CONNECTIONS

In this section we will analyze the implications of the soft-max surrogate loss in (5.6). Observe that the standard CRF surrogate loss is recovered by setting $\beta = 1$ and using a loss $\Delta_{\boldsymbol{y}}(\boldsymbol{y}') = 0\ \forall \boldsymbol{y}'$. We start our analysis by first considering the limit case of $\beta \to \infty$ which leads to a probabilistic interpretation of the structured SVM. We then derive the dual, which shows a joint regularization by entropy and margin.

5.2.1 *Structured SVMs as a Limit Case*

Lemma 5.1. *The standard maximum margin loss used in the structured SVMs is obtained for the choice $\beta \to \infty$.*

Proof. The structured SVM is derived as a limit case of $\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$ for $\beta \to \infty$ by adopting the log-sum-exp "trick", commonly used for stable numerical evaluation of partition functions. The key idea is to factor out the maximum contribution of the partition function. Denote by $\boldsymbol{y}^\star = \operatorname{argmax}_{\boldsymbol{y}'} \langle \boldsymbol{w}_\Delta, \psi_\Delta(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y}) \rangle$ the output with the largest score. Substituting into (5.7) yields

$$\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = \big\langle \boldsymbol{w}_\Delta, \psi_\Delta(\boldsymbol{y}^\star|\boldsymbol{x}, \boldsymbol{y}) \big\rangle + \frac{1}{\beta} \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp\left(\beta \Big(\big\langle \boldsymbol{w}_\Delta, \psi_\Delta(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y}) - \psi_\Delta(\boldsymbol{y}^\star|\boldsymbol{x}, \boldsymbol{y}) \big\rangle \Big) \right). \quad (5.9)$$

The second term becomes zero when $\beta \to \infty$, as the only terms in the sum that do not vanish, are outputs with exactly the same score as the maximum output $\boldsymbol{y}^*$. These terms evaluate to 1. Note that the number of maxima is independent of β. Hence, the soft-max loss for $\beta \to \infty$ results in:

$$\ell_\infty(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = \max_{\boldsymbol{y}' \in \mathcal{Y}} \big\langle \boldsymbol{w}_\Delta, \psi_\Delta(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{y}) \big\rangle .$$

which recovers the surrogate loss of the structured SVM in (5.1). Alternatively, the same result can be obtained using conjugate duality applied to the partition function, see Subsection 2.2.2, Section 2.4 and Subsection 2.7.3. □

A comparison of CRFs and structured SVMs reveals two important differences. First, the maximum-margin loss is only affected by the output that has the largest inner product. All the other outputs are do not influence the loss. Second, the loss $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$, which is not leveraged on at training time in the CRFs, provides a degree of freedom to specify how much loss a given output $\boldsymbol{y}'$ should incur given the ground-truth $\boldsymbol{y}$.

5.2.2 *Special Case: Binary Classification*

To illustrate the new surrogate loss, we discuss the special case of binary classification where $y \in \{-1, +1\}$. For binary classification,

as already discussed in Subsection 2.6.1, the feature map $\phi(x, y) = \frac{1}{2} y \phi(x)$, transforms the surrogate loss to

$$\ell_\beta(w, x, y) = \langle w, \phi(x, y) \rangle - \frac{1}{\beta} \log\Big(\exp\big(\beta \langle w, \phi(x, y) \rangle\big) + \exp\big(\beta(\langle w, \phi(x, y') \rangle + \Delta)\big) \Big).$$

Where y' denotes the wrong label $y' = -y$. The standard SVM emerges in the limit $\beta \to \infty$ and $\Delta = 1$. The parameter choice $\beta = 1$ and $\Delta = 0$ yields the Logistic Regression classifier. Different instantiations of this soft-max loss are visualized in Figure 5.1, including the log-loss and the max-margin loss.

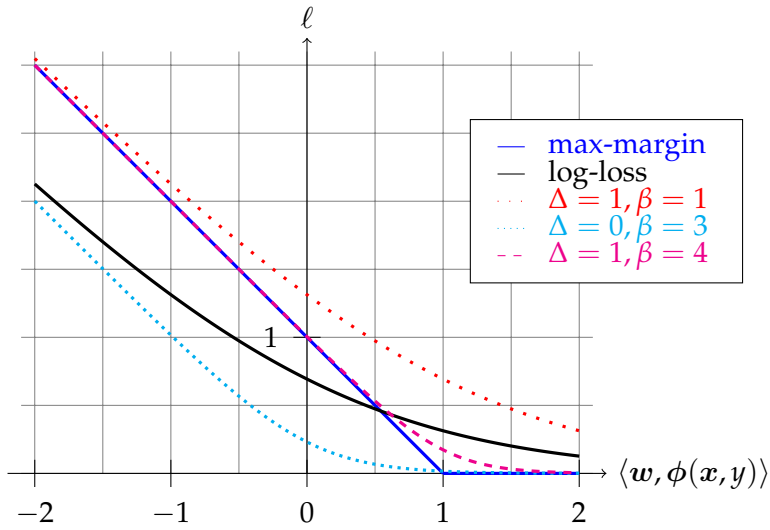


Figure 5.1: Soft-max surrogate loss $\ell_\beta(w, x, y)$ for different inverse temperatures β compared to log-loss and max-margin loss.

For the special case of binary classification, the influence of the inverse temperature on $\ell_\beta(w, x, y)$ was in parts discussed in (Zhang and Oles 2000). In our work we focus on classifiers for structured outputs. In this setting the effective number of negative outputs can be exponentially large, which makes the analysis more complex.

5.2.3 *Regularization by Entropy and Margin*

The dual of ERM with our new surrogate loss can be found by using the method of Lagrange, resulting in Lemma 5.2. The derivations are similar as in (Collins et al. 2008).

Lemma 5.2. *The dual minimization problem corresponding to (2.17) using our soft-max loss $\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$, is given by*

$$\begin{aligned} \max_{\boldsymbol{\mu} \in \mathbb{R}^{N \cdot |\mathcal{Y}|}} \quad & -\frac{1}{2\lambda} \boldsymbol{\mu}^\mathsf{T} \boldsymbol{K} \boldsymbol{\mu} + \boldsymbol{e}^\mathsf{T} \boldsymbol{\mu} - \frac{1}{\beta N} \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathcal{Y}} \mu_{n,\boldsymbol{y}} \log \mu_{n,\boldsymbol{y}} \\ \text{s.t.} \quad & \mu_{n,\boldsymbol{y}} \geq 0 \; \forall \boldsymbol{y} \in \mathcal{Y}, n \quad \text{and} \quad \sum_{\boldsymbol{y} \in \mathcal{Y}} \mu_{n,\boldsymbol{y}} = 1 \; \forall n \end{aligned} \tag{5.10}$$

where $\mu_{n,\boldsymbol{y}}$ denotes the dual variable for the output $\boldsymbol{y}$ in the n-th training example and $\boldsymbol{K}$ is given by $K_{(n,\boldsymbol{y}),(m,\boldsymbol{y}')} = \langle \frac{1}{N} \boldsymbol{\psi}_{n,\boldsymbol{y}}, \frac{1}{N} \boldsymbol{\psi}_{m,\boldsymbol{y}'} \rangle$. The difference between two mapped outputs is denoted by $\boldsymbol{\psi}_{n,\boldsymbol{y}} = -\boldsymbol{\psi}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n) = \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y})$. Furthermore, all the possible error terms are collected in a vector $\boldsymbol{e}$: $e_{n,\boldsymbol{y}} = \frac{1}{N} \Delta_{\boldsymbol{y}^n}(\boldsymbol{y})$. A total of $N \cdot |\mathcal{Y}|$ dual variables are required. The primal and dual variables are related by

$$\boldsymbol{w} = \frac{1}{\lambda N} \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathcal{Y}} \mu_{n,\boldsymbol{y}} \boldsymbol{\psi}_{n,\boldsymbol{y}}.$$

This connection between primal and dual variables holds for both, margin and slack rescaling. In the case of slack rescaling, $\boldsymbol{\psi}^{sl}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n)$ should be used instead of $\boldsymbol{\psi}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n)$.

Proof. The primal Lagrangian $\mathcal{L}(\boldsymbol{w})$ which is minimized w.r.t. $\boldsymbol{w}$ is given by

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{\beta} \log \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(\beta \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n) \rangle + \beta \Delta_{\boldsymbol{y}^n}(\boldsymbol{y})\right) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2.$$

We define $z_{n,\boldsymbol{y}} := \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n)\rangle + \Delta_{\boldsymbol{y}^n}(\boldsymbol{y})$ and scale the constraint by $1/N$, we thus get

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{z}) = &\frac{1}{N}\sum_{n=1}^{N}\frac{1}{\beta}\log\sum_{\boldsymbol{y}\in\mathcal{Y}}\exp(\beta z_{n,\boldsymbol{y}}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \\ &- \frac{1}{N}\sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathcal{Y}}\mu_{n,\boldsymbol{y}}[z_{n,\boldsymbol{y}} - \langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n)\rangle - \Delta_{\boldsymbol{y}^n}(\boldsymbol{y})] \\ = &-\frac{1}{N}\sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathcal{Y}} z_{n,\boldsymbol{y}}\mu_{n,\boldsymbol{y}} + \frac{1}{N}\sum_{n=1}^{N}\frac{1}{\beta}\log\sum_{\boldsymbol{y}\in\mathcal{Y}}\exp(\beta z_{n,\boldsymbol{y}}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \\ &+ \frac{1}{N}\sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathcal{Y}}\mu_{n,\boldsymbol{y}}[\langle \boldsymbol{w}, \boldsymbol{\psi}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n)\rangle + \Delta_{\boldsymbol{y}^n}(\boldsymbol{y})].\end{aligned}$$

Taking the derivatives w.r.t. the primal variable $\boldsymbol{w}$ we get

$$\frac{\partial\mathcal{L}(\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{z})}{\partial \boldsymbol{w}} = \lambda\boldsymbol{w} + \frac{1}{N}\sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathcal{Y}}\mu_{n,\boldsymbol{y}}\boldsymbol{\psi}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n),$$

and thus

$$\boldsymbol{w}^\star = -\frac{1}{\lambda N}\sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathcal{Y}}\mu_{n,\boldsymbol{y}}\boldsymbol{\psi}(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{y}^n)] = \frac{1}{\lambda N}\sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathcal{Y}}\mu_{n,\boldsymbol{y}}\boldsymbol{\psi}_{n,\boldsymbol{y}}.$$

The dual function is therefore

$$\begin{aligned}\inf_{\boldsymbol{z},\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{z}) = &-\frac{1}{2\lambda}\boldsymbol{\mu}^\mathsf{T}\boldsymbol{K}\boldsymbol{\mu} + \boldsymbol{e}^\mathsf{T}\boldsymbol{\mu} \\ &- \frac{1}{N}\sup_{\boldsymbol{z}}\left[\boldsymbol{\mu}^\mathsf{T}\boldsymbol{z} - \frac{1}{\beta}\sum_{n=1}^{N}\log\sum_{\boldsymbol{y}\in\mathcal{Y}}\exp(\beta z_{n,\boldsymbol{y}})\right] \\ = &-\frac{1}{2\lambda}\boldsymbol{\mu}^\mathsf{T}\boldsymbol{K}\boldsymbol{\mu} + \boldsymbol{e}^\mathsf{T}\boldsymbol{\mu} - \frac{1}{\beta N}\sum_{n=1}^{N}\sum_{\boldsymbol{y}\in\mathcal{Y}}\mu_{n,\boldsymbol{y}}\log\mu_{n,\boldsymbol{y}}.\end{aligned}$$

In addition, we get the constraints $\boldsymbol{\mu}_{n,\boldsymbol{y}} \geq 0$ and $\sum_{\boldsymbol{y}}\boldsymbol{\mu}_{n,\boldsymbol{y}} = 1$. The last step follows from the minimization w.r.t. $\boldsymbol{z}$:

$$\boldsymbol{\mu}_{n,\boldsymbol{y}} = \frac{\exp(\beta z_{n,\boldsymbol{y}})}{\sum_{\boldsymbol{y}'\in\mathcal{Y}}\exp(\beta z_{\boldsymbol{y}'})}.$$

This is solvable for $\mu_{n,\boldsymbol{y}} \geq 0$, $\sum_{\boldsymbol{y}}\mu_{n,\boldsymbol{y}} = 1$:

$$z^*_{n,\boldsymbol{y}} = \frac{1}{\beta}\log(\mu_{n,\boldsymbol{y}}).$$

This derivation is given in more detail in (Boyd and Vandenberghe 2004, Section 3.3.1). In its essence it reduces to identifying the entropy as the conjugate dual of the log partition function. Putting this back into the dual we get the final dual problem. □

The dual in (5.10) consists of three terms. The first term is similar to the kernel matrix in standard binary SVMs. The second term corresponds to a margin term which factors in the loss of the different output choices. The third term corresponds to the entropy of the distributions specified by $\boldsymbol{\mu}_n$. Contrary to log-loss and max-margin loss, both margin and entropy terms are present in the unified soft-max loss. Unsurprisingly, the log-loss and max-margin loss can also be identified as special cases in the dual: if $\boldsymbol{e}$ is the zero vector, we obtain the dual of the standard CRF, if $\beta \to \infty$ the dual of the structured SVM, see e.g. (Collins et al. 2008) for the dual formulations of the structured SVM and CRF.

5.2.4 *The Effect of the Inverse Temperature β*

So far, we argued that in order to reconstruct the log-loss from ℓ_β, the parameters $\beta = 1$ as well as a zero error term $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$ need to be used. However, the dual in (5.10) shows that it is actually sufficient to only alter the inverse temperature β and the regularization parameter λ, but not the error term itself. For a sufficiently small λ and β, the error term contribution $\boldsymbol{e}^\mathsf{T}\boldsymbol{\mu}$ becomes negligible compared to the first and third terms. As a result we identify the CRF dual.

As we have seen, β changes the peakedness of the conditional probability $P_\beta(\boldsymbol{y}'|\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})$. For $\beta \to 0$ all outputs $\boldsymbol{y}'$ have a uniform distribution, i.e., $P_\beta(\boldsymbol{y}'|\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})$ has an entropy of $\log(|\mathcal{Y}|)$. For $\beta \approx 1$ the distribution behaves similar to a CRF. For large values of β the probability mass concentrates on the outputs with the largest scores. Probabilities on the outputs are in this case not well-defined; the distribution consists of individual, scaled Dirac impulses at the outputs $\boldsymbol{y}^*$ with maximum scores. These findings are in line with (Bartlett and Tewari 2007), where SVMs are shown to be incapable of estimating conditional probabilities in a multiclass setting.

Lemma 5.3. *Let* $g_{mm}(\boldsymbol{w}) := \frac{1}{N}\sum_{n=1}^{N} \ell_{mm}(\boldsymbol{w}, \boldsymbol{x}^n, \boldsymbol{y}^n) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$ *and let* $g_\beta(\boldsymbol{w}) := \frac{1}{N}\sum_{n=1}^{N} \ell_\beta(\boldsymbol{w}, \boldsymbol{x}^n, \boldsymbol{y}^n) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$ *denote the ERM objective for the*

maximum margin and soft-max loss, respectively. The difference between the two objectives can then be bounded as follows:

$$g_{mm}(\boldsymbol{w}) \leq g_\beta(\boldsymbol{w}) \leq g_{mm}(\boldsymbol{w}) + \frac{1}{\beta} \log |\mathcal{Y}|.$$

In particular this also holds for $\boldsymbol{w}^\star = \min_{\boldsymbol{w}} g_\beta(\boldsymbol{w})$. *One therefore obtains an* additive approximation guarantee *of the error when training with the soft-max loss instead of the maximum margin loss.*

Proof. In order to derive an upper bound we start by expressing the soft-max loss through its dual. We use the formalism introduced in Section 2.4.

$$\begin{aligned}
\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{\beta} \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp\left(\beta(\langle \boldsymbol{w}, \psi(\boldsymbol{x}^n, \boldsymbol{y})\rangle) + \Delta_{\boldsymbol{y}}(\boldsymbol{y}'))\right). \\
&= \frac{1}{\beta} \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp(\beta \langle \bar{\boldsymbol{\theta}}, \varphi(\boldsymbol{y}')\rangle) = \frac{1}{\beta} \max_{\boldsymbol{\mu} \in \mathcal{M}} \left[\beta \langle \bar{\boldsymbol{\theta}}, \boldsymbol{\mu}\rangle + H(\boldsymbol{\mu})\right] \\
&\leq \max_{\boldsymbol{\mu} \in \mathcal{M}} \langle \bar{\boldsymbol{\theta}}, \boldsymbol{\mu}\rangle + \frac{1}{\beta} \log |\mathcal{Y}| = \ell_{mm}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) + \frac{1}{\beta} \log |\mathcal{Y}|.
\end{aligned}$$

Here $\bar{\boldsymbol{\theta}}$ denotes the score of a configuration according to $\boldsymbol{w}$ and the feature map. Substituting into $g_\beta(\boldsymbol{w})$ we obtain

$$g_\beta(\boldsymbol{w}) \leq g_{mm}(\boldsymbol{w}) + \frac{1}{\beta} \log |\mathcal{Y}|.$$

The lower bound trivially follows from the fact that the entropy is always positive. □

5.2.5 *Choosing* β

At this point it is natural to ask: "What is the best choice for β?" Ideally, β is optimized based on the training data. However, looking at the dual in (5.10), a model order selection question arises. By naively minimizing the surrogate loss w.r.t. β, this would always result in choosing $\beta \to \infty$, which is not desired. We thus advocate determining β via *cross validation* on hold out data. Instead of only performing cross validation over the regularization weight λ, a grid search for β as well as λ needs to be carried out.

As discussed above, one can think of the soft-max surrogate loss as an approximation of the structured SVM with more favorable convergence

for the overall optimization problem. Favorable convergence stems from the fact that the soft-max is smooth and therefore continuously differentiable. The soft-max surrogate loss can be understood as an application of the entropy smoothing approach of Nesterov (2005) to the maximum margin surrogate loss. In the context of learning structured models, it is interesting that applying smoothing to the structured SVMs identifies the CRF as a special case. In the context of inference, rather than learning, a similar smoothing approach has been used in (Jojic, Gould, and Koller 2010) to solve the LP relaxation efficiently.

In order to evaluate the partition function in the soft-max surrogate loss, marginal inference is required to be tractable, which might however be more difficult to solve than computing the MPE label, see Section 2.7. In some of the experiments we have found the soft-max loss to lead to better accuracy than the max-margin loss for a relatively small β. We attribute this to the fact that both losses are only approximations to the true loss. In challenging problems with a lot of overlap between the classes it seems to be beneficial that *several low-energy outputs influence the learning objective*, as opposed to only the output with the lowest energy, as it is the case in the structured SVM.

5.2.6 *Prediction*

So far this chapter has only considered the parameter estimation problem. We introduced a unified loss, parametrized by β, which is equivalent to the structured SVM and the CRF for limit cases of β. However, in the previous chapters we have advocated MPE for prediction in the structured SVM and minimum Bayes risk for the CRF. How should one predict in case the soft-max surrogate loss is used for learning? We will investigate this question in the experiments. As already discussed, we would not expect that minimum Bayes risk prediction would work well when $\boldsymbol{w}$ is estimated using a large β, as the estimated probabilities are no longer sensible.

5.3 LATENT VARIABLES

We now turn our attention to structured classifiers for partially observed data. As discussed in Section 2.5, two training objectives have been suggested for this more challenging setting: The Hidden Conditional

Random Field (HCRF) (Quattoni et al. 2007), and the latent structural SVM (Felzenszwalb, McAllester, and Ramanan 2008; Yu and Joachims 2009). Here we show that our formulation also extends to this scenario. Incorporating hidden variables into the output is an important extension of practical relevance: some outputs might be unobservable or one might define a hidden cause that leads to better accuracy of the predictions. Let us denote the observed output variables by $\boldsymbol{y}$ and the hidden, unobserved output variables (latent variables) by $\boldsymbol{z} \in \mathcal{Z}$. In HCRFs, the conditional probability of jointly observing $\boldsymbol{y}$ and $\boldsymbol{z}$ is modeled by the Gibbs distribution:

$$P_\beta(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z_\beta(\boldsymbol{x}, \boldsymbol{w})} \exp\Big(\beta\big\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big\rangle\Big).$$

Here, we directly include the inverse temperature β; $\beta = 1$ recovers the standard HCRF (Quattoni et al. 2007). In case of a zero-one loss, Bayesian risk minimization would predict according to

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \sum_{\boldsymbol{z} \in \mathcal{Z}} P_\beta(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}, \boldsymbol{w}).$$

Comparing this to the fully observed MPE prediction rule, we see that hidden variables are marginalized out. The introduction of the error terms into the Gibbs distribution by multiplying the two posterior distributions yields

$$P_\beta(\boldsymbol{y}', \boldsymbol{z}|\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) = \frac{1}{Z_\beta(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})} \exp\Big(\beta\big\langle \boldsymbol{w}_\Delta, \boldsymbol{\phi}_\Delta(\boldsymbol{y}', \boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})\big\rangle\Big).$$

with $\boldsymbol{\phi}_\Delta(\boldsymbol{y}', \boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}) = [\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{z})^\mathsf{T}, \Delta_{\boldsymbol{y}}(\boldsymbol{y}')]^\mathsf{T}$. Here it is assumed that $\Delta_{\boldsymbol{y}}(\boldsymbol{y}')$ is only dependent on observed output variables and not on the hidden variables. As in the CRF, training of the parameters is performed by minimizing the regularized negative log-likelihood, scaled by $1/\beta$. However, for the partially observed case, the hidden variables $\boldsymbol{z}$ have to be integrated out. This leads to the soft-max loss for latent variable settings:

$$\begin{aligned}\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{\beta} \log \sum_{\boldsymbol{z} \in \mathcal{Z}} \exp\Big(\beta\big\langle \boldsymbol{w}_\Delta, \boldsymbol{\phi}_\Delta(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})\big\rangle\Big) \\ + \frac{1}{\beta} \log \sum_{\substack{\boldsymbol{y}' \in \mathcal{Y} \\ \boldsymbol{z}' \in \mathcal{Z}}} \exp\Big(\beta\big\langle \boldsymbol{w}_\Delta, \boldsymbol{\phi}_\Delta(\boldsymbol{y}', \boldsymbol{z}'|\boldsymbol{x}, \boldsymbol{y})\big\rangle\Big).\end{aligned}$$

Taking the limit for $\beta \to \infty$ and using the log-sum-exp "trick", the latent structured SVM surrogate loss emerges:

$$\ell_\infty(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = -\max_{\boldsymbol{z} \in \mathcal{Z}} \langle \boldsymbol{w}_\Delta, \boldsymbol{\phi}_\Delta(\boldsymbol{y}, \boldsymbol{z} | \boldsymbol{x}, \boldsymbol{y}) \rangle + \max_{\substack{\boldsymbol{y}' \in \mathcal{Y} \\ \boldsymbol{z}' \in \mathcal{Z}}} \langle \boldsymbol{w}_\Delta, \boldsymbol{\phi}_\Delta(\boldsymbol{y}', \boldsymbol{z}' | \boldsymbol{x}, \boldsymbol{y}) \rangle.$$

Again, the latent structured SVM can be seen as a probabilistic model, in which all the probability mass is concentrated on the $\boldsymbol{y}, \boldsymbol{z}$ combination with the largest score. The limit case of the inverse temperature also changes the prediction for new test data to

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}, \boldsymbol{z} \in \mathcal{Z}} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle.$$

Instead of marginalizing the hidden variables out, we now maximize them out. The introduction of latent variables in general turns the empirical risk minimization in (2.17) into a non-convex optimization problem.

5.4 IMPLEMENTATION

So far we have focused on a theoretical comparison of the different surrogate losses for structured output learning. In this section we will discuss issues that are important for an actual implementation.

5.4.1 *Minimization of the Objective*

Our soft-max loss $\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$ is both convex (for the completely observed case) and differentiable for any inverse temperature except when $\beta \to \infty$, thus standard conjugate-gradient or L-BFGS solvers are applicable for the minimization of the surrogate loss. In our implementations we use L-BFGS. This choice is contrary to the minimization of the standard max-margin objective, where special algorithms for non-differentiable minimization problems are required. We refer to (Lacoste-Julien et al. 2013) for an overview of the different max-margin solvers. For learning with $\ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$, we are also interested in its derivative w.r.t. $\boldsymbol{w}$. For fully observed data and margin rescaling, the gradient takes a form similar to that of standard CRFs:

$$\frac{\partial \ell_\beta(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{w}} = -\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) + \sum_{\boldsymbol{y}' \in \mathcal{Y}} P_\beta(\boldsymbol{y}' | \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}'). \tag{5.11}$$

In our implementation we use the gradient information for the efficient minimization of the surrogate loss. The L-BFGS algorithm computes an approximation to the Hessian of the objective. For small β, this second-order information drastically improves the running time of the training. For large β, the Hessian does not help as the objective becomes essentially piecewise linear (assuming a small λ).

5.4.2 *Efficient Inference in Training*

One key step in the optimization of the objective function is the evaluation of the log-partition function $Z_\beta(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})$, which is generally computationally intractable. There exist cases, like for example a $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ that corresponds to a tree structured graphical model, where the computation of $Z_\beta(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})$ can be performed efficiently. Moreover, to compute the gradient in (5.11) the marginals also need to be efficiently computable, which is generally the case when the partition function is tractable. The structured SVM on the other hand requires computing the maximum violating output $\boldsymbol{y}^* = \operatorname{argmax}_{\boldsymbol{y}' \in \mathcal{Y}} \langle \boldsymbol{w}_\Delta, \boldsymbol{\phi}_\Delta(\boldsymbol{x}, \boldsymbol{y}') \rangle$. The maximization task in general is computationally hard, too, but there exist classes of problems where the maximization is tractable, but not the computation of the partition function. An example of such a case is when submodularity constraints are imposed on the potentials of a general graphical model. For more details, see Section 2.7 and Chapter 4.

5.5 RELATED WORK

Since (Zhang and Oles 2000), there have been various attempts to unify the max-margin and the log-loss. The connections between SVMs and exponential families have been indicated in (Canu and Smola 2006), and our work makes the link between the log-loss and max-margin loss more explicit through the inverse temperature and also extends to structured classifiers and latent variables. In (Chapelle and Zien 2005) an algorithm for learning multiclass SVMs in the primal is discussed: The max-margin loss is approximated by a soft-max, which can then be optimized by a conjugate-gradient solver. (Zhang 2005) considers a surrogate loss function similar to ours, applied to multiclass SVMs.

Independently, the softmax-margin loss was developed in (Gimpel and Smith 2010). The proposed surrogate loss and ours are very similar in spirit: both introduce the margin concept known from structured SVMs also into CRFs. In the application of named-entity recognition which they consider, the margin term shows to substantially improve the accuracy of the classifier. However, the connection between CRFs and structured SVMs is not established in their work. In another concurrent work Hazan and Urtasun (2010) studied the same soft-max loss function as in our work and also derived an algorithm for approximate learning.

5.6 EXPERIMENTS

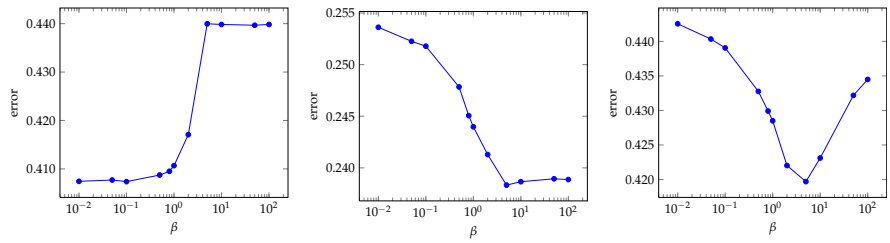
In our experiments we will only consider settings with either a small number of outputs $|\mathcal{Y}|$, or where inference can be performed exactly, such as scenarios where the feature map $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ corresponds to a chain structured graphical model.

5.6.1 *Multiclass Classification*

As a first experiment we consider the well-studied multiclass setting in which a data point is assigned to one of K classes. For a more detailed discussion of the model, see Subsection 2.6.2.

SYNTHETIC We designed three synthetic datasets with the reasoning in Subsection 5.2.5 in mind. Each of the datasets shows different characteristics, which can be exploited by the surrogate loss. The first dataset, Synth1, consists of three classes. Each class is sampled from a Gaussian with a mean at 0, 1 and 2, respectively and variance 1. We would expect a small β to perform best on this dataset, as the classes overlap to a large extent and probabilistic classifiers have been shown to perform well in such scenarios. The second dataset, Synth2, consists of three classes. Each class is sampled from a Gaussian with a mean at $(0, 0)$, $(1, 0.1)$ and $(1, -0.1)$, respectively, and with covariance $0.25\boldsymbol{I}$. Here, the prediction error is computed by accounting only 0.1 for a confusion between class 2 and 3, and 1 otherwise. This information is provided to the classifier using the loss term. We expect the best results with a large β, as the information of the loss is crucial. The third dataset, Synth3, consists of four Gaussian. Two of which have the mean

at $(0,0)$ and $(1,0)$, the remaining two have almost indistinguishable mean of $(0.5,0.4)$ and $(0.5,0.6)$. All classes have a covariance of $\boldsymbol{I}$. Again, for the almost indistinguishable classes we only account an error of 0.1 when confusing them. Here we would expect an intermediate value of β to lead to the best results, as both considerable class overlap and skewed class importance are present. The training set consists of 2000 examples for each class, the test set of 10000 examples for each class. For all classifiers $\lambda = 10^{-5}$ is fixed, as there is enough data to prevent overfitting. Similar results are obtained for different values of λ. The results of this experiment are shown in Figure 5.2. We observe



(a) Synth1 (b) Synth2 (c) Synth3

Figure 5.2: Test error for different classifiers obtained by learning using the soft-max with varying values of β on the synthetic multiclass datasets.

that the inverse temperature can have a substantial influence on the accuracy of the resulting classifier. No value of β is optimal for all three datasets, which is in agreement with the discussion in Subsection 5.2.5. The experiment also shows that the limit case of a SVM for $\beta \to \infty$ is already achieved for a relatively small β. We also tried prediction using the minimum Bayes risk predictor, leading to qualitatively similar accuracies.

MNIST For the MNIST dataset[1] we use the zero-one loss term in training and evaluation, as there is no structure on the class labels themselves. Minimum Bayes risk prediction and MPE coincide in this case, hence we only report one of the two numbers. We perform 5-fold cross-validation to determine the best combination of the hyper-parameters from the ranges $\lambda \in \{0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 50, 100\}$ and $\beta \in \{0.1, 1, 10, 100\}$. Once the best hyper-parameter is found using

1 We obtained the data from http://www.cs.nyu.edu/~roweis/data.html.

cross-validation, the model is re-trained on the combined training and validation data. If we perform full cross-validation over both hyper-parameters we obtain a test error of 7.1%, the same result was also obtained using the multi-class SVM implemented in liblinear. Figure 5.3 shows the results for different choices of β. We notice that low val-

β	λ	test error
0.1	1	7.37
1	5	7.23
10	50	7.10
100	50	7.12

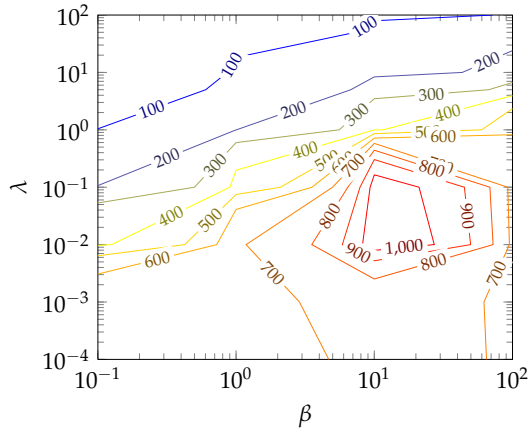


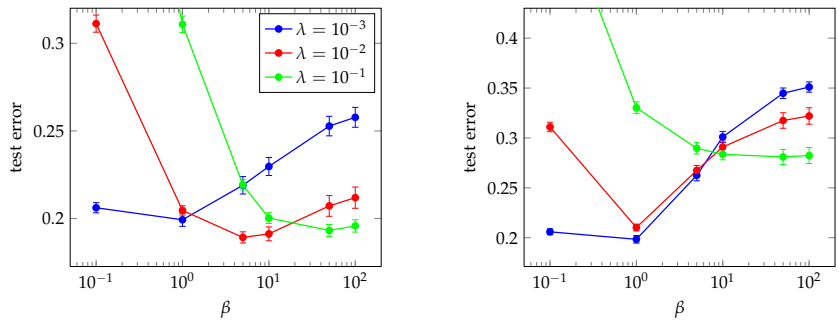
Figure 5.3: *Left*: Test error on the MNIST dataset for different values of the inverse temperature β. λ corresponds to the optimal regularizer weight found using cross-validation. *Right*: Training running time in seconds for different hyper-parameters on the MNIST dataset. As we would expect, learning gets more expensive for large β and small λ.

ues of β need little L_2-regularization, as the entropy term is already regularizing the model.

5.6.2 *Sequence Prediction*

In this experiment we consider the Optical Character Recognition (OCR) dataset from (Taskar, Guestrin, and Koller 2003). Here, the task is to predict the letters of a word from a given sequence of binary images. As has been shown in (Taskar, Guestrin, and Koller 2003), by exploiting the dependencies between neighboring letters, the accuracy of the classifier can be improved when compared to a model that classifies each letter independently. We use the same folds as in the original publication: The dataset consists of 10 train/test set splits, with each approximately 600 train and 5500 test sequences. We used the Hamming distance as

a loss term and perform inference in the linear chain model using the backward-forward algorithm implemented in UGM. In our experiments we obtained a test error of around 19% (Figure 5.4), slightly outperforming the structured SVM results reported in (Taskar, Guestrin, and Koller 2003). Varying the parameter β leads to a small, but consistent improvement over extreme values of β. We perform a second experiment on this



(a) MPE (b) MPM

Figure 5.4: OCR results for different values of β. On the left we show the test error when MPE prediction is used, on the right if MPM is used. The error bars show the standard deviation over the 10 different folds.

dataset to evaluate the quality of the probabilities on outputs learned by the model. To do so, we measure the error when predicting using the Maximum Posteriori Marginal (MPM) instead of the MPE predictor. MPM is the optimal predictor according to Bayesian decision theory for the true posterior distribution over outputs. Using the MPM leads to good accuracy for small values of β, but fails for larger β. This behaviour is in agreement with our discussion in Subsection 5.2.4 that probabilities on outputs are not well-defined for the structured SVM.

5.6.3 *Multiple Instance Learning*

As a last experiment we consider the problem of learning from multiple instances, see Subsection 2.6.6. Multiple instance learning is a scenario with latent variables in training, as the label of an individual instance in a bag is not observed; only the label of the whole bag. The model for $\beta = 1$ and no loss term recovers the Multiple Instance Logistic

Regression from (Ray and Craven 2005), for $\beta \to \infty$ the model reduces to the MI-SVM (Andrews, Tsochantaridis, and Hofmann 2002).

We construct a one-dimensional synthetic dataset which illustrates the deficiencies of the MI-SVM. A positive bag consists of p positive instances and $50 - p$ negative ($0 < p \leq 50$), a negative bag contains 50 negative instances. The individual instances are hard to classify: the positive instances are Gaussian distributed with mean 0.6 whereas the negative instances are Gaussian distributed with mean 0, the variance for both classes is 1. Smaller values of β lead to better classification performance, as this corresponds to an averaging over the different instances in a bag, which is a good strategy for large data uncertainty, see Figure 5.5.

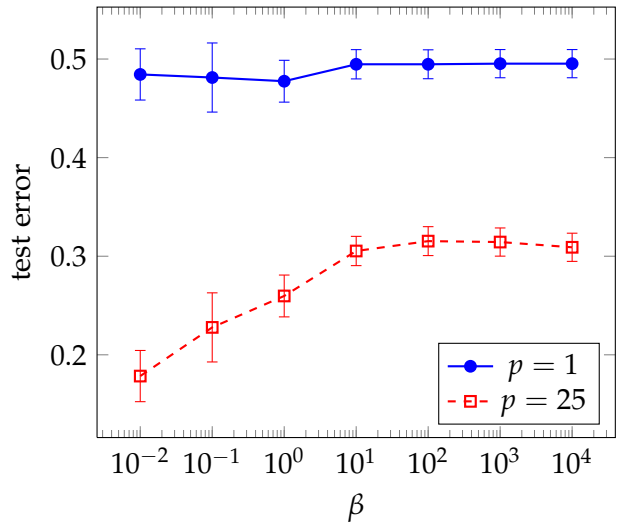


Figure 5.5: Results for the synthetic multiple instance learning dataset for 800 bags, averaged over 8 random datasets. Depending on the number p of positive instances, a small β improves the accuracy substantially. The solid line corresponds to a setting where only one instance per bag is positive, the dashed line to 25 positive instances per bag.

5.7 DIRECT LOSS MINIMIZATION

So far the present chapter studied structured output learning from a surrogate loss minimization point of view. In reality one would however

like to *directly minimize the loss of a structured predictor*. As we will show here, a modification of the soft-max loss proposed in this chapter also applies to this more challenging setting. Let us denote the *score* of an input/output variable pair by $s_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y})$. We study two different scores, corresponding to the score used in structured SVMs and Bayesian risk minimization.

1. Linear score:
$$s^l_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) := \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle.$$

2. Negative expected risk:
$$\begin{aligned} s^r_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) &:= \log \left(\sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp\Big(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}') \rangle + \log(1 - \Delta_{\boldsymbol{y}'}(\boldsymbol{y})) \Big) \right) \\ &= \log \left(\sum_{\boldsymbol{y}' \in \mathcal{Y}} (1 - \Delta_{\boldsymbol{y}'}(\boldsymbol{y})) \exp(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}') \rangle) \right) \\ &\propto \log \left(\sum_{\boldsymbol{y}' \in \mathcal{Y}} (1 - \Delta_{\boldsymbol{y}'}(\boldsymbol{y})) P(\boldsymbol{y}' | \boldsymbol{x}, \boldsymbol{w}) \right). \end{aligned}$$
Here $P(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{w})$ denotes the standard conditional Gibbs distribution of a CRF.

A prediction function is then obtained by returning the output with the largest score for a given input:
$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} s_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}).$$

Our goal here is to choose the parameters $\boldsymbol{w}$ directly so that the empirical loss $\sum_{n=1}^{N} \Delta_{\boldsymbol{y}^n}(f_{\boldsymbol{w}}(\boldsymbol{x}^n))$ is minimized. For now we ignore regularization of $\boldsymbol{w}$. The main problem with direct loss minimization arises from the fact that the objective is difficult to optimize, as it is non-convex and piece-wise constant. A recent non-convex surrogate loss, the *ramp loss*, was initially proposed in the context of binary SVMs (Collobert et al. 2006) and later extended to structured classifiers (Do et al. 2008), also see (McAllester and Keshet 2011). It is defined as[2]:
$$\ell_{\alpha,ramp}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = \max_{\boldsymbol{y}' \in \mathcal{Y}} \left[\alpha s_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}') + \Delta_{\boldsymbol{y}}(\boldsymbol{y}') \right] - \max_{\boldsymbol{y}' \in \mathcal{Y}} \alpha s_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}'). \quad (5.12)$$

2 Note that there exist different definitions of the ramp loss in the literature, we adopt the one in (McAllester and Keshet 2011).

Here $\alpha \in \mathbb{R}^+$ is a strictly positive scalar and we already adopted the score notation. The ramp loss, to the best of our knowledge, has so far only been studied in the context where the score is given by $s_{\boldsymbol{w}}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle$, but not for prediction functions that are based on Bayesian risk minimization. If $s_{\boldsymbol{w}}(\boldsymbol{x})$ is convex in $\boldsymbol{w}$, which is the case for both scores studied, also the two terms in the definition of the ramp loss are convex, this follows from the fact that convexity is preserved by the maximum operation. Hence the objective is a difference of convex functions, which is in general a non-convex objective, to which however Concave-Convex Procedure (CCCP) is immediately applicable. CCCP (Yuille and Rangarajan 2003) is a general approach to solve non-convex optimization problems that can be expressed as a difference of convex functions, also see Subsection 3.4.1 for more details about the CCCP algorithm.

An important property of the ramp loss is that it approaches the true loss:

$$\lim_{\alpha \to \infty} \ell_{\alpha,ramp}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = \Delta_{\boldsymbol{y}}(f_{\boldsymbol{w}}(\boldsymbol{x})).$$

This limit follows from the fact that for a large α, the first and second maximizing arguments in (5.12) are the same and their score therefore cancels out. Only the true loss of the output chosen according to the prediction function $f_{\boldsymbol{w}}(\boldsymbol{x})$ remains.

The soft-max idea is also applicable to the ramp loss. By replacing the two maximum functions by a log-sum-exp approximation, we obtain a *soft ramp loss*:

$$\ell_{\alpha,\beta}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\beta} \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp(\beta(\alpha s_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}') + \Delta_{\boldsymbol{y}}(\boldsymbol{y}'))) - \frac{1}{\beta} \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp(\beta \alpha s_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}')).$$

In the limit case $\alpha \to \infty$ and $\beta \to \infty$ the soft ramp loss results in the direct loss:

$$\lim_{\alpha \to \infty, \beta \to \infty} \ell_{\alpha,\beta}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = \Delta_{\boldsymbol{y}}(f_{\boldsymbol{w}}(\boldsymbol{x})).$$

An advantage of the soft ramp loss compared to the standard ramp loss, stems from its differentiability, which could potentially lead to improved numerical solvers. The benefits could be two-fold. First, as the smoothness of the objective can be adapted by changing β, it should be possible to obtain algorithms with better convergence guarantees.

Second, the smoothing could potentially mitigate the problem of getting stuck in poor local minima.

If the soft ramp loss is convexified by replacing the second partition function by the inner product of the ground-truth configuration and the score $s_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle$ is used, the approach reduces to the soft-max for $\alpha = 1$.

5.8 CONCLUSIONS

We have introduced a novel family of surrogate losses for structured output learning. The soft-max loss is parametrized by an inverse temperature β which controls the entropy of the posterior distribution on outputs. The dual of the surrogate loss shows a double regularization by a margin and an entropy term. The max-margin loss and the log-loss emerge as two special cases of this loss. Additionally, our work also extends to models with hidden variables and direct loss minimization. We conjecture that different applications require different values of β and validate this claim experimentally on multiclass, linear-chain models and multiple instance learning. Choosing a large β, which corresponds to a large margin setting, while sometimes improving the accuracy, shows to have the severe disadvantage of deteriorating the probability distribution on outputs. The difference between the surrogate losses for different values of β is particularly striking in the multiple instance learning experiment.

6

PART & CLAMP

This chapter considers the problem of learning the parameters of a CRF and formulates a novel *lower bound* on the intractable partition function. The evaluation of the bound typically requires only a few iterations of a modified message passing algorithm, where the number of iterations is fixed and dependent on the specified budget. Our approach consists of two parts: First, a subset of the output variables is selected, a so called Feedback Vertex Set (FVS), with the property that any cycle in the graph contains at least one variable in the FVS. Second, a conditioned partition function for one state of the variables in the FVS is repeatedly computed for a few low-energy states, to successively tighten the lower bound. Each individual computation requires two message passes. Let $\mathcal{D}$ denote a set of these FVSs and $\mathcal{Z}$ the states of the variables in the FVS. We show that the lower bound, which we denote by $Z(\boldsymbol{x}, \boldsymbol{w}, \mathcal{D}, \mathcal{Z})$, an exact definition will be given later, is well-suited for structured output learning with a minimum negative log-likelihood objective[1] (see Subsection 2.5.1):

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \left[-\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}^n) \rangle + \log Z(\boldsymbol{x}^n, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}^n) \right]. \tag{6.1}$$

The contributions of this chapter are as follows: First, we propose a generalization of composite likelihood for computing a lower approximation of the structured partition function and formulate a tightening strategy. We show that composite likelihood is a specific instance of this framework. Second, we introduce a forest decomposition and formulate it as a minimal FVS problem. Third, a variational algorithm, MAX-TIGHTEN is introduced. The algorithm finds the states of the FVS which maximally increase the lower bound. We introduce batch and online algorithms for learning with the lower bound. The performance of the algorithms is demonstrated on a computer vision dataset.

1 For the sake of clarity we will omit regularization terms in this chapter; the experiments are performed including a regularizer.

6.1 LOWER BOUNDING THE STRUCTURED OUTPUT LOSS

This section introduces an extension of composite likelihood (see Subsection 2.8.1) which can be understood as a lower bound of the partition function. The goal in this work is to approximate an intractable partition function of the form

$$Z(\boldsymbol{x}, \boldsymbol{w}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle).$$

Extensions to partition functions as encountered in Chapter 5, with an inverse temperature or a difference of feature maps are straightforward. For the sake of notational convenience we restrict ourselves to the standard form of the structured partition function. As in composite likelihood, we assume a partition of $\mathcal{V}$ into two sets, $\mathcal{A}$ and $\mathcal{B}$ is given.

partition function

The partition function can be decomposed into two sums running over the states of the variables in $\mathcal{A}$ and $\mathcal{B}$. A trivial lower-bound is obtained by summing over only a (small) subset $\underline{\mathcal{Y}}_{\mathcal{B}} \subseteq \mathcal{Y}_{\mathcal{B}}$ of the large state space $\mathcal{Y}_{\mathcal{B}}$:

$$\begin{aligned} Z(\boldsymbol{x}, \boldsymbol{w}) &= \sum_{\boldsymbol{y}_{\mathcal{B}} \in \mathcal{Y}_{\mathcal{B}}} \sum_{\boldsymbol{y}_{\mathcal{A}} \in \mathcal{Y}_{\mathcal{A}}} \exp(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle) \\ &\geq \sum_{\boldsymbol{y}_{\mathcal{B}} \in \underline{\mathcal{Y}}_{\mathcal{B}}} \sum_{\boldsymbol{y}_{\mathcal{A}} \in \mathcal{Y}_{\mathcal{A}}} \exp(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle) \\ &=: Z(\boldsymbol{x}, \boldsymbol{w}, \mathcal{B}, \underline{\mathcal{Y}}_{\mathcal{B}}). \end{aligned}$$

The set $\mathcal{Y}_{\mathcal{B}}$ contains all possible states of the variables in $\mathcal{B}$ and is therefore exponentially large. Using a subset $\underline{\mathcal{Y}}_{\mathcal{B}}$ may result in a relatively poor approximation for high-entropy distributions. However, as we will show, for parameter learning this simple approach can be very effective. The choice of the decomposition, $(\mathcal{A}, \mathcal{B})$ as well as the states in the set $\underline{\mathcal{Y}}_{\mathcal{B}}$ are discussed in detail in Subsection 6.2.1 and Subsection 6.2.2 respectively. Actual learning algorithms are given in Subsection 6.2.3. The remainder of this section discusses extensions of the lower bound and its connection to previous work.

6.1.1 *Several Decompositions*

In order to decrease the effects of poor decompositions, several partitions $\mathcal{D} = \{(\mathcal{A}_1, \mathcal{B}_1), \ldots, (\mathcal{A}_M, \mathcal{B}_M)\}$ and corresponding states $\mathcal{Z} =$

$\{\underline{\mathcal{Y}}_{\mathcal{B}_1}, \ldots, \underline{\mathcal{Y}}_{\mathcal{B}_M}\}$ can be combined. The *arithmetic and geometric mean as well as the maximum of all the bounds are also valid lower bounds*:

$$
\begin{aligned}
Z^{\mathrm{a}}(\boldsymbol{x}, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}) &:= \frac{1}{M} \sum_{m=1}^{M} Z(\boldsymbol{x}, \boldsymbol{w}, \mathcal{B}_m, \underline{\mathcal{Y}}_{\mathcal{B}_m}), \\
Z^{\mathrm{g}}(\boldsymbol{x}, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}) &:= \left(\prod_{m=1}^{M} Z(\boldsymbol{x}, \boldsymbol{w}, \mathcal{B}_m, \underline{\mathcal{Y}}_{\mathcal{B}_m}) \right)^{1/M}, \\
Z^{\mathrm{m}}(\boldsymbol{x}, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}) &:= \max_{m} Z(\boldsymbol{x}, \boldsymbol{w}, \mathcal{B}_m, \underline{\mathcal{Y}}_{\mathcal{B}_m}).
\end{aligned}
\tag{6.2}
$$

The maximum over the different decompositions results in the tightest bound, but has the disadvantage of being non-differentiable w.r.t. the parameters due to the maximum function. This discontinuity can cause problems when minimizing the (regularized) negative log-likelihood in (6.1) using a quasi-Newton solver, such as L-BFGS that rely on the smoothness of the objective. One approach to deal with the non-differentiability would be to use a soft-max instead of the maximum. The geometric mean is commonly used by composite likelihood approaches, and is obtained by considering the arithmetic average of the log partition function. The geometric mean has the advantage that it is smooth. However, the arithmetic mean actually provides a tighter lower bound while maintaining differentiability.

Lemma 6.1. *The relation between the different lower bound combinations is*

$$
Z(\boldsymbol{x}, \boldsymbol{w}) \geq Z^{m}(\boldsymbol{x}, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}) \geq Z^{a}(\boldsymbol{x}, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}) \geq Z^{g}(\boldsymbol{x}, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}).
$$

While in terms of the *value* of the approximation, the maximum combination is preferable, this in not necessarily the case for learning with a lower bound: The quality of an approximation depends on the difference between the exact maximum likelihood solution and the minimizer obtained when using the lower bound as an approximation for the exact partition function as in (6.1).

6.1.2 *Connection to Composite Likelihood*

The following lemma draws the connection between the lower bound in (6.2) and composite likelihood.

Lemma 6.2. *Composite likelihood learning with decompositions $\mathcal{D}$ is equivalent to lower bounding the partition function in the negative log-likelihood for each example $(\boldsymbol{x}^n, \boldsymbol{y}^n)$ by the geometric average bound $Z^g(\boldsymbol{x}^n, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}^n)$ with $\mathcal{Z}^n = \{\underline{\mathcal{Y}}^n_{\mathcal{B}_1}, \ldots, \underline{\mathcal{Y}}^n_{\mathcal{B}_M}\}$ where $\underline{\mathcal{Y}}^n_{\mathcal{B}_m} = \{\boldsymbol{y}^n_{\mathcal{B}_m}\}$.*

Proof. We choose to scale the composite likelihood objective by the number of decompositions M. Assuming the same number of decompositions is used for each example, this does not change the value of the minimizer. The composite likelihood contribution of an example $(\boldsymbol{x}, \boldsymbol{y})$ is

$$\begin{aligned}
&\frac{1}{M}\sum_{m=1}^{M} -\log P(\boldsymbol{y}_{\mathcal{A}_m}|\boldsymbol{y}_{\mathcal{B}_m}, \boldsymbol{x}, \boldsymbol{w}) \\
&\quad = \frac{1}{M}\sum_{m=1}^{M} \left[-\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\rangle + \log Z(\boldsymbol{y}_{\mathcal{B}_m}, \boldsymbol{x}, \boldsymbol{w})\right] \\
&\quad = -\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\rangle + \frac{1}{M}\sum_{m=1}^{M} \log Z(\boldsymbol{y}_{\mathcal{B}_m}, \boldsymbol{x}, \boldsymbol{w}).
\end{aligned}$$

Which recovers the geometric average lower bound. □

The lower bound in our work differs in two key aspects from the classic composite likelihood: First, we give a concrete choice of the decompositions by feedback-vertex sets, balancing computational tractability and accuracy of the estimator. Second, in addition to the ground-truth, several other low energy states of the variables in $\mathcal{B}$ are used. As we will show in the experiments, especially the second contribution improves the results drastically.

6.1.3 *Asymptotic Consistency*

The bound in (6.2), when used in a maximum likelihood objective, can be understood as an efficient extension of the non-local contrastive objective introduced in (Vickrey, Lin, and Koller 2010). In their work a learning objective is formulated where the partition function is replaced by an exhaustive enumeration over low energy states which are computed using MPE inference. Contrary to our work, the non-local contrastive objective does however not consider the decomposition of the partition function into two parts. Therefore many additional states need to be considered explicitly. Nevertheless, the proof of asymptotic

consistency in (Vickrey, Lin, and Koller 2010) also generalizes to our lower bound (when the geometric combination is used) subject to the same regularity assumptions.

6.1.4 *Comparison to Upper Bounds*

In contrast to our lower bound a considerable amount of work has investigated upper bounding the partition function. Upper bounds are generally obtained using variational inference. Such an example is the tree reweighted belief propagation (Wainwright, Jaakkola, and Willsky 2005a). A problem of learning with upper bounds such as (Hazan and Urtasun 2010; Wainwright, Jaakkola, and Willsky 2005a; Meshi et al. 2010) arises from the fact that convergence of the message passing algorithms used to compute the upper bound is generally slow, or sometimes not even guaranteed.

The general perception in the field is that upper bounds are superior to lower bounds, as intuitively speaking "less things can go wrong" when minimizing an upper bound. This view is also supported by (Finley and Joachims 2008), see the brief discussion in Subsection 2.8.3. Our work questions this belief by showing that composite likelihood, an often employed approach, is in fact a lower bound. Furthermore, we give experimental support that our lower bound leads to state-of-the-art accuracy on a well-studied data set.

6.1.5 *Connection to Cutset Conditioning*

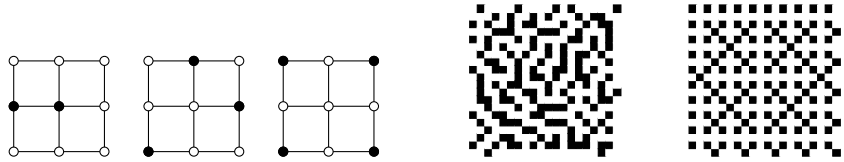
Our work is related to the relatively old idea of cutset conditioning in Bayesian networks, dating back to Pearl (1990), see also (Koller and Friedman 2009, Section 9.5). For the task of probabilistic inference, a subset of the nodes is exhaustively enumerated over, whereas for the remaining variables a sum-product algorithm is used. We use a very similar idea in the next section. Cutset conditioning is generalized in (Horvitz, Suermondt, and Cooper 1989) for approximate inference. However, to the best of our knowledge, cutset conditioning has not been used for learning undirected models, nor have its connection to composite likelihood been explored.

6.2 PART & CLAMP

This section describes the details of the main building blocks of our proposed *part & clamp* algorithm.

6.2.1 *Part: Finding a Minimum FVS*

The lower bound in (6.2) is valid for any choice of partition $\mathcal{A}, \mathcal{B}$. For tractable computations, we consider $\mathcal{B}$ such that all loops in the graph $\mathcal{G}$ are blocked by at least one variable in $\mathcal{B}$. Such a subset of the output variables is called a feedback vertex set (Vazirani 2001). To make this property explicit, we will use $\mathcal{F}$ to denote such a set $\mathcal{B}$ and $\mathcal{V} \backslash \mathcal{F}$ to denote its complement. Due to the FVS property (see Figure 6.1), $\mathcal{V} \backslash \mathcal{F}$ is a forest and hence conditioned on the state of the FVS, $\boldsymbol{y}_{\mathcal{F}}$, summation over $\mathcal{Y}_{\mathcal{V} \backslash \mathcal{F}}$ for the remaining variables can be carried out exactly using the sum-product algorithm (see Subsection 2.7.2).



(a) example FVSs for 3×3 grid (b) DF (0.38) (c) BF (0.33)

Figure 6.1: Different feedback vertex sets (in black) of a grid-graph. (b) & (c) show small FVSs obtained using the algorithms described below. BF and DF denote the breadth-first and depth-first approach. We indicate the fraction of variables in the FVS in brackets (the smaller the better). A checkerboard decomposition would have a fraction of around 0.5.

The decision variant of the minimal FVS is an NP-hard problem (Vazirani 2001) and therefore one has to resort to approximation algorithms. In our work we consider the unweighted version, where the number of variables $|\mathcal{F}|$ in the FVS is minimized. This can be motivated by the principle of insufficient reason: all the variables are assumed to have the same contribution to the partition function. Therefore, the minimum FVS results in the lowest approximation error. More complex selection criteria based on marginal variable weights would be possible. However, in our work we *keep the decomposition fixed during learning* and

thus an initial estimate of the parameters would need to be obtained in order to make the selection based on marginals meaningful.

Algorithm 6.1 Growing forests algorithm for the FVS problem.

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
$\mathcal{F} = \emptyset, \mathcal{Q} = \emptyset, \forall i \in \mathcal{V} : \text{visited}(i) = 0$.
while not all vertices visited **do**
  Choose i at random from $\{j \in \mathcal{V} : \text{visited}(j) = 0\}$.
  $i \rightarrow Q$.
  repeat
    $i \leftarrow Q$.
    if $|\{j \in \mathcal{N}(i) : \text{visited}(j) \wedge j \notin \mathcal{F}\}| \geq 2$ **then**
      $\mathcal{F} = \mathcal{F} + \{i\}$.
    end if
    $\text{visited}(i) = 1$.
    $\forall j \in \mathcal{N}(i) \wedge \neg\text{visited}(j) : j \rightarrow Q$ (random order).
  until $Q = \emptyset$.
end while
return $\mathcal{F}$

A series of papers (Becker and Geiger 1996; Chudak et al. 1998; Bafna, Berman, and Fujito 1999) gives 2-approximation algorithms for the minimum FVS problem. In our work we consider a simpler probabilistic algorithm (Chandrasekaran et al. 2011) based on a breadth-first exploration of the graph. The algorithm is shown in Algorithm 6.1; $\mathcal{N}(i)$ denotes the neighborhood of a vertex i. An algorithm with a depth-first exploration is obtained by using a stack instead of the queue $\mathcal{Q}$. Generally the results with depth-first exploration were inferior to the ones obtained using breadth-first. For a grid graph the breadth-first approach leads to a close to optimal FVS ratio of around $1/3$, which is in the order of the lower bound in (Luccio 1998).

6.2.2 *Clamp: Choosing the States of the FVS*

The set $\underline{\mathcal{Y}}_{\mathcal{F}}$ is initialized with the ground-truth label $\boldsymbol{y}^n_{\mathcal{F}}$, corresponding to the input $\boldsymbol{y}^n$ as the only state. The learning algorithm which we will describe later in more detail, estimates $\boldsymbol{w}$ for a given bound and proceeds by adding one additional state to $\underline{\mathcal{Y}}_{\mathcal{F}}$. This alternating procedure is repeated till convergence. The tightening procedure described below will therefore be run for *different weight vectors* and hence the maximiz-

ing argument will most likely differ between iterations. For a given parameter $\boldsymbol{w}$ and feedback vertex set $\mathcal{F}$ let us consider the problem of finding a labeling $\boldsymbol{y}_{\mathcal{F}}^{\star}$ to be included in $\underline{\mathcal{Y}}_{\mathcal{F}}$. We here follow a greedy approach by including $\boldsymbol{y}_{\mathcal{F}}^{\star}$ to *maximally tighten* the lower bound in (6.2). We choose the states of the FVS according to:

$$\boldsymbol{y}_{\mathcal{F}}^{\star} = \operatorname*{argmax}_{\boldsymbol{y}_{\mathcal{F}} \in \mathcal{Y}_{\mathcal{F}}} \sum_{\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}} \in \mathcal{Y}_{\mathcal{V}\setminus\mathcal{F}}} \exp(\langle \boldsymbol{w}, \phi(\boldsymbol{x}, \boldsymbol{y}) \rangle). \tag{6.3}$$

The maximization above is more complex than standard energy minimization problems arising from MPE inference. The task is sometimes described as marginal MPE (see Section 2.7). Here, some variables $\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}$ are summed over and other variables $\boldsymbol{y}_{\mathcal{F}}$ are maximized over. We derive a variational approach for this problem, which we named MAX-TIGHTEN. To simplify the notation, we drop the dependence of the energy on $\boldsymbol{w}$ and $\boldsymbol{x}$. Furthermore, let $Z(\boldsymbol{y}_{\mathcal{F}})$ denote the partition function for the FVS variables clamped to state $\boldsymbol{y}_{\mathcal{F}}$.

We follow the recent approach in (Liu and Ihler 2011; Jiang, Rai, and Daumé III 2011) which formulates the marginal MPE as a variational problem over the marginal polytope (see Subsection 2.7.3). Let us first rewrite the log partition function for a given $\boldsymbol{y}_{\mathcal{F}}$ through its dual:

$$\begin{aligned} A(\boldsymbol{y}_{\mathcal{F}}) &:= \log \sum_{\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}} \exp(-E(\boldsymbol{y})) \\ &= \min_{P} \sum_{\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}} P(\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}|\boldsymbol{y}_{\mathcal{F}}) E(\boldsymbol{y}) \\ &\quad + \sum_{\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}} P(\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}|\boldsymbol{y}_{\mathcal{F}}) \log P(\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}|\boldsymbol{y}_{\mathcal{F}}) \end{aligned}$$

The full problem in (6.3) can then be rewritten as

$$\max_{\boldsymbol{y}_{\mathcal{F}}} A(\boldsymbol{y}_{\mathcal{F}}) = \min_{P} \sum_{\boldsymbol{y}} P(\boldsymbol{y}) E(\boldsymbol{y}) + \underbrace{\sum_{\boldsymbol{y}} P(\boldsymbol{y}) \log P(\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}|\boldsymbol{y}_{\mathcal{F}})}_{=:-H(\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}|\boldsymbol{y}_{\mathcal{F}})}. \tag{6.4}$$

The first term in (6.4) is a standard average energy and the second term corresponds to the negative *conditional entropy* of $\boldsymbol{y}_{\mathcal{V}\setminus\mathcal{F}}$. Unfortunately, the variational problem above still remains intractable as the optimization problem has an exponential number of variables (or equivalently an exponential number of constraints if expressed using the marginal polytope). Furthermore, the conditional entropy does not factorize into

marginals, which makes an approximation even more difficult. Both (Liu and Ihler 2011) and (Jiang, Rai, and Daumé III 2011) choose to approximate (6.4) by relaxing the constraint set to the local marginal polytope and replacing the entropy term with a unary and pairwise approximation. We here choose the approach from (Jiang, Rai, and Daumé III 2011) which reduces to a hybrid message-passing algorithm in which for variables in the FVS a max-product update is performed, whereas for the remaining nodes a sum-product update is used.

For obtaining a $\boldsymbol{y}^\star_{\mathcal{F}}$, the obtained pseudo-beliefs need to be rounded to integer values. We choose to round each node independently to the state with the largest pseudo-belief.

Our approach ignores the constraint that $\boldsymbol{y}^\star_{\mathcal{F}}$ should be different from all the states already in $\underline{\mathcal{Y}}_{\mathcal{F}}$ and thus in theory might generate a state that is already modeled. This could potentially be improved with an approach akin to the M-best MAP algorithm (Fromer and Globerson 2009), at the cost of an increased runtime. We observed empirically that states were rarely chosen repeatedly.

Alternatively, instead of using the MAX-TIGHTEN approach, one can simply compute a MPE labeling and use it to tighten the bound. While this approach does not guarantee the most effective tightening, it has the advantage that very efficient specialized solvers are available. In practice we have found that initializing the message-passing algorithm for MAX-TIGHTEN with a smoothed version of a MPE label consistently lead to the best results[2].

6.2.3 *Derived Learning Algorithms*

Here we describe parameter learning with the proposed lower bound. The approximate learning objective is obtained by replacing the partition function in the negative log-likelihood by $Z(\boldsymbol{x}, \boldsymbol{w}, \mathcal{D}, \mathcal{Z})$, leading to the objective given in (6.1):

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \left[-\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}^n) \rangle + \log Z(\boldsymbol{x}^n, \boldsymbol{w}, \mathcal{D}, \mathcal{Z}) \right].$$

To make learning practical, the gradient of the approximate objective needs to be computed efficiently. The derivative of the approximate log

2 A smoothed version is obtained by setting the pseudo-beliefs of a variable to $1 - \alpha$ for the MPE state and to $\alpha/(K-1)$ for the $K-1$ other states.

partition function w.r.t. $\boldsymbol{w}$ is the expected feature map. The expectation now only runs over the states in the individual $\underline{\mathcal{Y}}_{\mathcal{F}}$ for each decomposition. The expectation computation requires the marginals $P_\alpha(\boldsymbol{y}_\alpha|\underline{\mathcal{Y}}_{\mathcal{F}})$ of the factors for each decomposition. If only one decomposition is used, the gradient is given by

$$\frac{\partial}{\partial \boldsymbol{w}} = \frac{1}{N}\left[\sum_{n=1}^{N} -\boldsymbol{\phi}(\boldsymbol{x}^n, \boldsymbol{y}^n) + \sum_{t\in\mathcal{T}} \sum_{\alpha\in\mathcal{C}(t)} \sum_{\boldsymbol{y}_\alpha\in\mathcal{Y}_\alpha} P_\alpha(\boldsymbol{y}_\alpha|\underline{\mathcal{Y}}^n_{\mathcal{F}})\boldsymbol{\phi}_t(\boldsymbol{x}^n, \boldsymbol{y}_\alpha, \alpha)\right].$$

All the required quantities can be computed by aggregating the results from simple sum-product runs for the different clamping configurations $\boldsymbol{y}_{\mathcal{F}} \in \underline{\mathcal{Y}}_{\mathcal{F}}$. We illustrate this fact for two configurations $\boldsymbol{y}^1_{\mathcal{F}}, \boldsymbol{y}^2_{\mathcal{F}}$; the case of larger $\underline{\mathcal{Y}}_{\mathcal{F}}$ is straightforward. Given the marginals $P_\alpha(\boldsymbol{y}_\alpha|\boldsymbol{y}^1_{\mathcal{F}})$ and $P_\alpha(\boldsymbol{y}_\alpha|\boldsymbol{y}^2_{\mathcal{F}})$ for the two clamping states and the corresponding partition functions Z^1, Z^2, the combined quantities are obtained as follows:

$$\begin{aligned} Z^{1,2} &= Z^1 + Z^2, \\ P_\alpha(\boldsymbol{y}_\alpha|\underline{\mathcal{Y}}_{\mathcal{F}}) &= \frac{Z^1_\alpha(\boldsymbol{y}_\alpha)}{Z^{1,2}} + \frac{Z^2_\alpha(\boldsymbol{y}_\alpha)}{Z^{1,2}}, \\ Z^k_\alpha(\boldsymbol{y}_\alpha) &= Z^k P_\alpha(\boldsymbol{y}_\alpha|\boldsymbol{y}^k_{\mathcal{F}}) \quad \text{for } k \in \{1,2\}. \end{aligned} \tag{6.5}$$

As described in Subsection 6.1.1, several decompositions can be handled using different combination approaches. The marginal computations for the maximum and geometric average combinations are simple (marginals of the decomposition with the maximum value and an average of the different marginals, respectively). The arithmetic average combination turns out to be a partition function computation of a special form, which is however also tractable.

BATCH LEARNING Batch learning for the proposed lower bound computes M decompositions of the graphical model for each example. The set $\underline{\mathcal{Y}}^n_{\mathcal{F}}$ for each of the examples $\boldsymbol{x}^n$ is initialized with the ground-truth label $\boldsymbol{y}^n_{\mathcal{F}}$. The resulting bound is then minimized using L-BFGS leading to a first parameter estimate. For this parameter, a tightening operation using MAX-TIGHTEN is performed for each decomposition and example. The tightening of the lower bound is followed by a minimization w.r.t. the parameters. These steps are repeated until convergence. The batch algorithm has close relationships to the cutting planes algorithm employed in the training of structured SVMs (Tsochantaridis et al. 2005). Also, if a pseudo-likelihood decomposition is used together

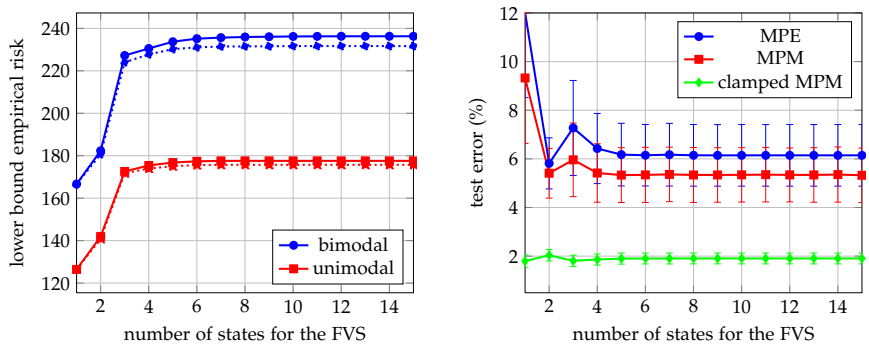
with the soft-max loss (see Chapter 5), for the limit case of an infinite inverse temperature, the pseudo-max approach introduced in (Sontag et al. 2010) is recovered.

ONLINE LEARNING In order to solve large-scale problems, we propose an online learning version of the algorithm using Stochastic Gradient Descent (SGD)(Robbins and Monro 1951; Kiefer and Wolfowitz 1952). SGD evaluates the loss for a subset of examples and takes a step in the direction of the gradient. In our implementation, a budget is specified for each example, this budget corresponds to the number of states in $\underline{\mathcal{Y}}^n_{\mathcal{F}}$. At each iteration the lower bound is tightened using MAX-TIGHTEN and in case the budget is exceeded, the highest energy state is pruned from $\underline{\mathcal{Y}}^n_{\mathcal{F}}$ (the ground-truth state is however never removed). Followed by an evaluation of the lower bound and its derivative.

6.3 EXPERIMENTS

We evaluate the performance of the proposed approach on the application of binary image denoising. We use the dataset in (Kumar and Hebert 2006) and follow their experimental settings. The dataset consists of two noise scenarios: a unimodal and bimodal noise model. 10 images are used for training and 150 images for testing. The graphical model is given by the standard four connected grid of size 64×64. In all of the experiments we use $\lambda = 1$, we observed little change when varying the regularization parameter. Figure 6.2 shows the development of the lower bound for batch learning. As expected, the lower bound increases as more labels are added to $\underline{\mathcal{Y}}_{\mathcal{F}}$. Furthermore, the more states considered for the FVS, the better the test error. Comparing the curves for the unimodal and bimodal noise datasets, we notice that the lower bound flattens off after fewer iterations of the algorithm in the easier unimodal dataset. For MPE prediction we used Sequential Tree-Reweighted Message Passing (TRWS) (Kolmogorov 2006) and for MPM we used Gibbs sampling (with 1000 sweeps).

Table 6.1 shows the image denoising results obtained using different learning and prediction algorithms. As expected, MPM prediction outperforms MPE prediction. We also include a prediction version ('clamped MPM') where the minimal FVS algorithm is used to find a FVS,



(a) Empirical risk.

(b) Prediction error.

Figure 6.2: Batch learning for the binary image denoising dataset. The lower bound on the empirical risk and the prediction error are visualized when increasing $\underline{\mathcal{Y}}_{\mathcal{F}}$. We observe that the first couple of iterations are the most important. The dotted curve in (a) corresponds to tightening using MPE inference, the solid curve to MAX-TIGHTEN: The differences are more pronounced for the bimodal data set. The curves in (b) correspond to different prediction approaches using the same parameter estimate. The error bars show the standard deviation of the prediction errors. Note that clamped MPM is unrealistic (since it requires labels), and is shown to indicate the theoretical optimum.

the state of which is clamped to its *true value*[3]. Only for the variables in $\mathcal{V}\backslash\mathcal{F}$ a label is predicted. We report the test error of the clamped MPM to give a rough estimate of the prediction error underlying the surrogate loss used in composite likelihood training. As can be seen, the Part&Clamp approach, in both its online and batch version, performs well and does not exhibit the poor behavior of contrastive divergence on the more difficult bimodal dataset. The online version of Part&Clamp performs better than the batch version, as it seems to overfit less to the training data (the training error is however worse). The results improve slightly on (Kumar and Hebert 2006) where a regularization heuristic for pseudo-likelihood was used. (Hazan and Urtasun 2010) studies a different setting, which should have a diminishing test error

3 'Clamped MPM' considers an unrealistic scenario, as it assumes the value of the FVS variables are known even for the test examples.

	Prediction \ Train	Pseudo-likelihood	Composite likelihood	Contrastive divergence	Part & Clamp batch	Part & Clamp online
bimodal	MPE	15.58 ± 4.11	12.02 ± 3.50	7.01 ± 1.71	6.14 ± 1.27	$\mathbf{5.16 \pm 0.77}$
	MPM	11.86 ± 3.40	9.33 ± 2.69	6.72 ± 1.67	5.32 ± 1.12	$\mathbf{5.20 \pm 0.80}$
	clamped MPM	$\mathbf{1.77 \pm 0.25}$	1.80 ± 0.26	1.96 ± 0.22	1.90 ± 0.22	2.23 ± 0.25
unimodal	MPE	5.28 ± 1.47	4.43 ± 1.26	$\mathbf{2.39 \pm 0.47}$	2.40 ± 0.50	2.40 ± 0.46
	MPM	4.13 ± 1.18	3.66 ± 0.96	$\mathbf{2.37 \pm 0.45}$	2.40 ± 0.42	2.42 ± 0.43
	clamped MPM	0.98 ± 0.22	1.01 ± 0.21	$\mathbf{1.05 \pm 0.21}$	1.03 ± 0.22	1.17 ± 0.23

Table 6.1: Test error for learning on the binary image denoising dataset. A single FVS decomposition per example is used, i.e. $M = 1$. Composite likelihood refers to the first iteration of the Part&Clamp algorithm (i.e. only the ground-truth state is used in $\underline{\mathcal{Y}}_{\mathcal{F}}$). For the SGD updates in contrastive divergence and the online variant of Part&Clamp, all the images were considered for a single update step, making it a gradient descent step. This was done to simplify the comparison to the batch learning algorithms. For contrastive divergence five Gibbs iterations were used, for the online version of Part&Clamp a budget of two labels, i.e. $|\underline{\mathcal{Y}}_{\mathcal{F}}| = 2$.

as the output label is always the same in training and testing and the parametrization is powerful enough to simply remember this particular output label.

Figure 6.3 reports results when several decompositions are considered. In each setting the batch algorithm is run for one to five different FVSs with a clamping set of size five, i.e. $|\underline{\mathcal{Y}}^n_{\mathcal{F}}| = 5$. MPM inference is used for prediction. It can be observed that increasing the number of FVSs has little influence on the results. The combination approach (arithmetic or geometric mean) behaved as predicted by Lemma 6.1; the arithmetic mean leads to slightly more robust results. This behavior is however only visible if random states are used for clamping.

6.4 SUMMARY

We propose a lower approximation of the partition function which improves with increasing computational resources. Our method consists of finding good partitions of the graphical model (Part) and for those partitions find good states of the conditioning set (Clamp). We solve the first problem by finding a minimal FVS to obtain the largest

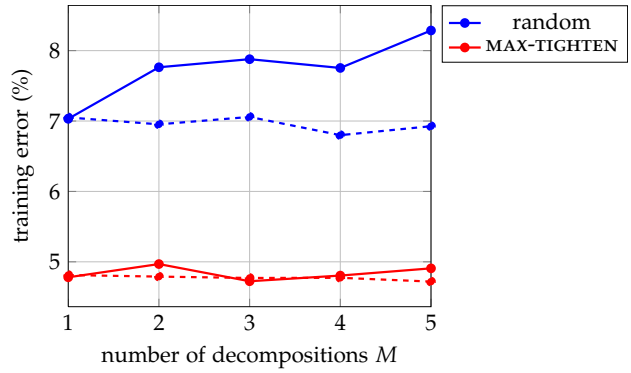


Figure 6.3: Training error for different number of decompositions for a clamping set of size five. We visualize the results when either using the max-tighten algorithm for tightening (red) or a random state for clamping (blue). The results with the geometric mean (solid line) are slightly less robust w.r.t. the decompositions than the arithmetic mean (dashed line) for the random clamping.

possible tractable subgraph. Then we propose a variational approach MAX-TIGHTEN to optimize the states of the conditioning set. We demonstrate that the resulting learning algorithm has good performance in a computer vision task. Furthermore, the online version should enable large-scale learning of conditional random fields.

7 DISCUSSION

The methods introduced in this thesis can be extended in various ways and are applicable to several new domains. Future work could potentially consider the following directions:

- As outlined in Chapter 5, the soft-max loss is not restricted to surrogate loss minimization, but is also applicable in a direct loss minimization scenario. We have not studied such an approach yet. We would expect the soft-max, due to its implicit regularization by the entropy, to lead to a smoother optimization landscape than approaches that rely on a maximum operation. Direct loss minimization holds considerable promise for further accuracy improvements of structured classifiers (Do et al. 2008).

- We argue that many vision problems, especially in medical imaging, are in fact counting tasks. The segmentation is often only an intermediate step, the final task is however to count the number of pixels that are for example infected. We have not fully explored these applications in our work and would expect the label-count loss, introduced in Chapter 4, to have a large impact in such scenarios. For medical applications it would also make sense to consider loss terms that are relating the number of infected pixels to a global staining label, which roughly corresponds to the amount of stained pixels and takes values in a predefined range. Further, the label-count loss is also applicable to computer vision applications, where one is given bounding-boxes in training, but would like to predict a pixel-accurate segmentation at test time. In such a scenario, a label-count approach could learn a structured classifier such that for example 75% of the pixels inside every bounding-box in an image are labeled as foreground. This would remain computationally tractable even with many bounding boxes in a single image.

- For the Part&Clamp work we focused on forest-like decompositions because the resulting sub-problems are exactly solved by the

sum-product algorithm. The approach is however not restricted to such tree graphs, and could potentially be extended to planar subgraphs (Schraudolph and Kamenetsky 2009), as this sub-problem is also exactly solvable in some situations.

- One interesting aspect of our work on the combined LPQP relaxation is the fact that the message-passing algorithm from a practical point becomes more efficient if the weight of the consistency constraint is increased. Future work should investigate this rather counter-intuitive trade-off of speed and accuracy in more detail. Our work also shows that in settings where the LP relaxation is not tight, solving an entropy-augmented LP relaxation to optimality for very small entropy weights, is not advisable. As we showed in Chapter 3, better results are obtained in such a setting by increasing in a second stage the influence of the entropy, combined with an appropriate modification of the unary terms. It would be desirable to derive a criteria based on which one can obtain the precision up to which the initial entropy-augmented LP relaxation should be solved.

- We suspect that one could include a similar constraint as in the LPQP objective when learning structured SVMs. Initially a variant of the structured SVM would be learned for a LP relaxation, which is successively tightened in subsequent iterations.

Approximate inference and learning has seen a wide-spread research interest and adoption in the past couple of years, as witnessed by several workshops at NIPS or ICML, but also more applied conferences such as CVPR or ICCV. Many novel algorithms are introduced every year, each work demonstrating the performance of their approach on a few example applications. Unfortunately, it is very difficult to *compare* the different approaches in an unbiased manner. This is only partially the fault of the researchers, as applying a structured algorithm to an application in practice is time-consuming due to domain-specific feature extraction or unclear modeling assumptions and hence with a restricted time-budget only few experiments can be performed. It is my strong personal belief that in order to advance the field as a whole, in a community effort a *benchmark for structured output prediction and learning* needs to be established. Such a benchmark should define a structured learning and prediction application programming interface together

with a standard file format. A website would allow researchers to obtain a large amount of structured problems, for which the algorithms can then be run. Different categories could be established, such as pairwise or higher-order graphical models, models with many states per variable, tractable and intractable models, or different parameterizations for learning. Such a benchmark would facilitate a better comparison of the different approaches found in the literature and allow the field to better understand the trade-offs between speed and accuracy. Furthermore, it would facilitate a closer interaction with applied researchers, who could upload novel problems to the website. Efforts such as the Probabilistic Inference Challenge (PIC) are steps in this direction, but should be extended along the direction of mldata.org.

BIBLIOGRAPHY

Andrew, Galen and Jianfeng Gao (2007). „Scalable training of L1-regularized log-linear models." In: *Proceedings of the 24th international conference on Machine learning (ICML)*. ACM, pp. 33–40 (cit. on p. 29).

Andrews, Stuart, Ioannis Tsochantaridis, and Thomas Hofmann (2002). „Support Vector Machines for Multiple-Instance Learning." In: *Advances in Neural Information Processing Systems 15 (NIPS)*, pp. 561–568 (cit. on pp. 39, 119).

Asuncion, Arthur U., Qiang Liu, Alexander Ihler, and Padhraic Smyth (2010). „Learning with Blocks: Composite Likelihood and Contrastive Divergence." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 9, pp. 33–40 (cit. on p. 56).

Auer, Peter (1997). „On Learning From Multi-Instance Examples: Empirical Evaluation of a Theoretical Approach." In: *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, pp. 21–29 (cit. on p. 39).

Bafna, Vineet, Piotr Berman, and Toshihiro Fujito (1999). „A 2-Approximation Algorithm for the Undirected Feedback Vertex Set Problem." In: *SIAM Journal on Discrete Mathematics* 12.3, pp. 289–297 (cit. on p. 129).

Bakir, Gökhan, Thomas Hofmann, Bernhard Schölkopf, Alex J. Smola, Ben Taskar, and S. V. N. Vishwanathan (2007). *Predicting Structured Data*. MIT Press (cit. on p. 7).

Bartlett, Peter L. and Ambuj Tewari (2007). „Sparseness vs Estimating Conditional Probabilities: Some Asymptotic Results." In: *Journal of Machine Learning Research* 8, pp. 775–790 (cit. on p. 109).

Beck, Amir and Marc Teboulle (2009). „A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems." In: *SIAM Journal on Imaging Sciences* 2.1, pp. 183–202 (cit. on p. 73).

Becker, Ann and Dan Geiger (1996). „Optimization of Pearl's method and greedy-like approximation algorithms for the vertex feedback set problem." In: *Artificial Intelligence* 83.1, pp. 167–188 (cit. on p. 129).

Bertsekas, D P (1999). *Nonlinear Programming*. Belmont, MA: Athena Scientific (cit. on p. 71).

Besag, Julian (1974). „Spatial Interaction and the Statistical Analysis of Lattice Systems." In: *Journal of the Royal Statistical Society. Series B (Methodological)* 36.2, pp. 192–236 (cit. on pp. 2, 17).

Besag, Julian (1975). „Statistical Analysis of Non-Lattice Data." In: *The Statistician* 24.3, pp. 179–195 (cit. on p. 54).

Bishop, Christopher M (2006). *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag (cit. on p. 47).

Blake, Andrew and Andrew Zisserman (1987). *Visual reconstruction*. Cambridge, MA, USA: MIT Press (cit. on p. 75).

Blake, Andrew, Carsten Rother, M. Brown, Patrick Pérez, and Philip H. S. Torr (2004). „Interactive Image Segmentation Using an Adaptive GMMRF Model." In: *8th European Conference on Computer Vision (ECCV)*, pp. 428–441 (cit. on pp. 49, 95).

Boros, Endre and Peter L. Hammer (2002). „Pseudo-Boolean optimization." In: *Discrete Applied Mathematics* 123.1-3, pp. 155–225 (cit. on p. 91).

Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik (1992). „A training algorithm for optimal margin classifiers." In: *Proceedings of the fifth annual workshop on computational learning theory (COLT)*, pp. 144–152 (cit. on p. 33).

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge University Press (cit. on p. 109).

Boyd, Stephen, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein (2011). „Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." In: *Foundations and Trends in Machine Learning* 3.1, pp. 1–122 (cit. on p. 75).

Boykov, Yuri (2001). „Fast approximate energy minimization via graph cuts." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8.4, pp. 413–1239 (cit. on pp. 49, 92).

Boykov, Yuri and Vladimir Kolmogorov (2004). „An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9, pp. 1124–1137 (cit. on p. 92).

Brand, Matthew and Patrick Pletscher (2008). „A conditional random field for automatic photo editing." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. viii).

Canu, Stéphane and Alex J. Smola (2006). „Kernel methods and the exponential family.“ In: *Neurocomputing* 69.7-9, pp. 714–720 (cit. on p. 114).

Chandrasekaran, Karthekeyan, Richard Karp, Erick Moreno-Centeno, and Santosh Vempala (2011). „Algorithms for Implicit Hitting Set Problems.“ In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 614–629 (cit. on p. 129).

Chandrasekaran, Venkat, Nathan Srebro, and Prahladh Harsha (2008). „Complexity of Inference in Graphical Models.“ In: *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 70–78 (cit. on p. 43).

Chapelle, Olivier and Alexander Zien (2005). „Semi-Supervised Classification by Low Density Separation.“ In: *Proceedings of the International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pp. 57–64 (cit. on p. 114).

Chudak, Fabián A., Michel X. Goemans, Dorit S. Hochbaum, and David P. Williamson (1998). „A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs.“ In: *Operations Research Letters* 22.4, pp. 111–118 (cit. on p. 129).

Collins, Michael, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett (2008). „Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks.“ In: *Journal of Machine Learning Research* 9, pp. 1775–1822 (cit. on pp. 107, 109).

Collobert, Ronan, Fabian Sinz, Jason Weston, and Léon Bottou (2006). „Trading Convexity for Scalability.“ In: *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. ACM, pp. 201–208 (cit. on p. 120).

Cooper, Gregory F. (1990). „The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks.“ In: *Artificial Intelligence* 42, pp. 393–405 (cit. on p. 43).

Cortes, Corinna and Vladimir Vapnik (1995). „Support-Vector Networks.“ In: *Machine Learning* 20.3, pp. 273–297 (cit. on p. 33).

Crammer, Koby and Yoram Singer (2002). „On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines.“ In: *Journal of Machine Learning Research* 2.2, pp. 265–292 (cit. on pp. 34, 35).

Dietterich, Thomas G., Richard H. Lathrop, and Tomás Lozano-Pérez (1997). „Solving the multiple instance problem with axis-parallel rectangles." In: *Artificial Intelligence* 89.1–2, pp. 31–71 (cit. on p. 39).

Dillon, Joshua V. and Guy Lebanon (2010). „Stochastic Composite Likelihood." In: *Journal of Machine Learning Research* 11, pp. 2597–2633 (cit. on p. 54).

Do, Chuong B., Quoc Le, Choon Hui Teo, Olivier Chapelle, and Alex Smola (2008). „Tighter Bounds for Structured Estimation." In: *Advances in Neural Information Processing Systems 21 (NIPS)*, pp. 281–288 (cit. on pp. 120, 137).

Domke, Justin (2011). „Dual Decomposition for Marginal Inference." In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (cit. on pp. 71, 72).

Duchi, John, Daniel Tarlow, Gal Elidan, and Daphne Koller (2006). „Using Combinatorial Optimization within Max-Product Belief Propagation." In: *Advances in Neural Information Processing Systems 19 (NIPS)*, pp. 369–376 (cit. on p. 94).

Elidan, Gal, Ian McGraw, and Daphne Koller (2006). „Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing." In: *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence (UAI)* (cit. on p. 47).

Everingham, Mark, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman (2010). „The Pascal Visual Object Classes (VOC) Challenge." In: *International Journal of Computer Vision* 88.2, pp. 303–338 (cit. on p. 84).

Felzenszwalb, Pedro F., David A. McAllester, and Deva Ramanan (2008). „A discriminatively trained, multiscale, deformable part model." In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 32, 112).

Felzenszwalb, Pedro F, Ross B Girshick, David McAllester, and Deva Ramanan (2010). „Object detection with discriminatively trained part-based models." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9, pp. 1627–1645 (cit. on pp. 32, 39).

Finley, Thomas and Thorsten Joachims (2008). „Training structural SVMs when exact inference is intractable." In: *Proceedings of the 25th international conference on Machine learning (ICML)*, pp. 304–311 (cit. on pp. 31, 35, 56, 127).

Fromer, Menachem and Amir Globerson (2009). „An LP View of the M-best MAP problem.“ In: *Advances in Neural Information Processing Systems 22 (NIPS)*, pp. 567–575 (cit. on p. 131).

Fujishige, Satoru (1991). *Submodular functions and optimization*. Amsterdam: Annals of Discrete Mathematics (cit. on p. 49).

Geman, Stuart and Donald Geman (1984). „Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6.6, pp. 721–741 (cit. on pp. 2, 17).

Gimpel, Kevin and Noah A. Smith (2010). „Softmax-Margin CRFs: Training Log-Linear Models with Cost Functions.“ In: *Human Language Technologies (HLT)*, pp. 733–736 (cit. on p. 115).

Goldberg, Andrew V., Sagi Hed, Haim Kaplan, Robert E. Tarjan, and Renato F. Werneck (2011). „Maximum Flows by Incremental Breadth-First Search.“ In: *Algorithms - ESA 2011 - 19th Annual European Symposium*, pp. 457–468 (cit. on p. 92).

Gould, Stephen (2011). „Max-margin Learning for Lower Linear Envelope Potentials in Binary Markov Random Fields.“ In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 193–200 (cit. on p. 88).

Gulshan, Varun, Carsten Rother, Antonio Criminisi, Andrew Blake, and Andrew Zisserman (2010). „Geodesic Star Convexity for Interactive Image Segmentation.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3129–3136 (cit. on p. 95).

Hammersley, John M. and Peter Clifford (1971). *Markov fields on finite graphs and lattices* (cit. on pp. 2, 17).

Hazan, Tamir and Amnon Shashua (2010). „Norm-Product Belief Propagation: Primal-Dual Message-Passing for Approximate Inference.“ In: *IEEE Transactions on Information Theory* 56.12, pp. 6294–6316 (cit. on pp. 47, 70).

Hazan, Tamir and Raquel Urtasun (2010). „A Primal-Dual Message-Passing Algorithm for Approximated Large Scale Structured Prediction.“ In: *Advances in Neural Information Processing Systems 23 (NIPS)* (cit. on pp. 115, 127, 134).

Hinton, Geoffrey E. (2002). „Training Products of Experts by Minimizing Contrastive Divergence.“ In: *Neural Computation* 14.8, pp. 1771–1800 (cit. on p. 55).

Hofmann, Thomas and Joachim M. Buhmann (1997). „Pairwise Data Clustering by Deterministic Annealing." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, pp. 1–14 (cit. on pp. 53, 76).

Horvitz, Eric, H. Jacques Suermondt, and Gregory F. Cooper (1989). „Bounded Conditioning: Flexible Inference for Decisions Under Scarce Resources." In: *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)* (cit. on p. 127).

Ising, Ernst (1925). „Beitrag zur Theorie des Ferromagnetismus." In: *Zeitschrift für Physik A Hadrons and Nuclei* 31.1, pp. 253–258 (cit. on pp. 2, 15).

Jaggi, Martin, Simon Lacoste-Julien, Mark Schmidt, and Patrick Pletscher (2012). „Block-Coordinate Frank-Wolfe for Structural SVMs." In: *NIPS Workshop on Optimization for Machine Learning* (cit. on p. viii).

Jancsary, Jeremy and Gerald Matz (2011). „Convergent decomposition solvers for tree-reweighted free energies." In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 15, pp. 388–398 (cit. on p. 64).

Jegelka, Stefanie, Hui Lin, and Jeff A. Bilmes (2011). „On fast approximate submodular minimization." In: *Advances in Neural Information Processing Systems 24 (NIPS)*, pp. 460–468 (cit. on p. 50).

Jiang, Jiarong, Piyush Rai, and Hal Daumé III (2011). „Message-Passing for Approximate MAP Inference with Latent Variables." In: *Advances in Neural Information Processing Systems 24 (NIPS)*, pp. 1197–1205 (cit. on pp. 130, 131).

Jojic, Vladimir, Stephen Gould, and Daphne Koller (2010). „Accelerated dual decomposition for MAP inference." In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pp. 503–510 (cit. on pp. 74, 111).

Kappes, Jörg H. and Christoph Schnörr (2008). „MAP-Inference for Highly-Connected Graphs with DC-Programming." In: *Pattern Recognition, 30th DAGM Symposium* (cit. on pp. 61, 65).

Kiefer, Jack and Jacob Wolfowitz (1952). „Stochastic estimation of the maximum of a regression function." In: *The Annals of Mathematical Statistics* 23, pp. 462–466 (cit. on pp. 29, 55, 133).

Kirkpatrick, Scott, C. Daniel Gelatt, and Mario P. Vecchi (1983). „Optimization by simulated annealing." In: *Science* 220, pp. 671–680 (cit. on pp. 53, 76).

Kohli, Pushmeet and M. Pawan Kumar (2010). „Energy Minimization for linear Envelope MRFs.“ In: *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1863–1870 (cit. on pp. 18, 50, 84, 89, 90).

Kohli, Pushmeet, M. Pawan Kumar, and Philip H. S. Torr (2009). „P^3 and Beyond: Solving Energies with Higher Order Cliques.“ In: vol. 31. 9, pp. 1645–1656 (cit. on pp. 18, 89).

Kohli, Pushmeet, Lubor Ladicky, and Philip H. S. Torr (2009). „Robust Higher Order Potentials for Enforcing Label Consistency.“ In: *International Journal of Computer Vision* 82.3, pp. 302–324 (cit. on p. 91).

Kolev, Kalin and Daniel Cremers (2008). „Integration of Multiview Stereo and Silhouettes Via Convex Functionals on Convex Domains.“ In: *10th European Conference on Computer Vision (ECCV)*, pp. 752–765 (cit. on p. 89).

Koller, Daphne and Nir Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press (cit. on pp. 7, 20, 43, 127).

Kolmogorov, Vladimir (2006). „Convergent tree-reweighted message passing for energy minimization.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10, pp. 1568–1583 (cit. on pp. 74, 76, 133).

Kolmogorov, Vladimir and Ramin Zabih (2004). „What Energy Functions Can Be Minimized via Graph Cuts?“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2, pp. 147–159 (cit. on pp. 51, 91).

Komodakis, Nikos, Nikos Paragios, and Georgios Tziritas (2007). „MRF optimization via dual decomposition: Message-passing revisited.“ In: *IEEE 11th International Conference on Computer Vision (ICCV)* (cit. on p. 71).

Krähenbühl, Philipp and Vladlen Koltun (2011). „Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials.“ In: *Advances in Neural Information Processing Systems 24 (NIPS)*, pp. 109–117 (cit. on p. 37).

Kschischang, Frank R., Brendan J. Frey, and Hans-Andrea Loeliger (2001). „Factor graphs and the sum-product algorithm.“ In: *IEEE Transactions on Information Theory* 47.2, pp. 498–519 (cit. on p. 11).

Kulesza, Alex and Fernando Pereira (2008). „Structured learning with approximate inference.“ In: *Advances in Neural Information Processing Systems 20 (NIPS)*, pp. 785–792 (cit. on p. 53).

Kumar, Akshat and Shlomo Zilberstein (2011). „Message-Passing Algorithms for Quadratic Programming Formulations of MAP Estimation.“ In: *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 428–435 (cit. on pp. 61, 65, 75, 80).

Kumar, Akshat, Shlomo Zilberstein, and Marc Toussaint (2012). „Message-Passing Algorithms for MAP Estimation Using DC Programming.“ In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 22, pp. 656–664 (cit. on pp. 62, 65, 75).

Kumar, Sanjiv and Martial Hebert (2006). „Discriminative Random Fields.“ In: *International Journal of Computer Vision* 68.2, pp. 179–201 (cit. on pp. 133, 134).

Lacoste-Julien, Simon, Martin Jaggi, Mark Schmidt, and Patrick Pletscher (2013). „Block-Coordinate Frank-Wolfe Optimization for Structural SVMs.“ In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. To appear (cit. on pp. viii, 85, 113).

Ladicky, Lubor (2011). „Global Structured Models towards Scene Understanding.“ PhD thesis. Oxford Brookes University (cit. on p. 37).

Ladicky, Lubor, Chris Russell, Pushmeet Kohli, and Philip H. S. Torr (2010). „Graph cut based inference with co-occurrence statistics.“ In: *Proceedings of the 11th European conference on Computer vision: Part V*. ECCV'10. Springer-Verlag, pp. 239–253 (cit. on p. 36).

Lafferty, John, Andrew McCallum, and Fernando Pereira (2001). „Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.“ In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pp. 282–289 (cit. on pp. 3, 21, 27).

Lan, Xiangyang, Stefan Roth, Daniel P. Huttenlocher, and Michael J. Black (2006). „Efficient Belief Propagation with Learned Higher-Order Markov Random Fields.“ In: *9th European Conference on Computer Vision (ECCV)*, pp. 269–282 (cit. on p. 83).

Lempitsky, Victor S. and Andrew Zisserman (2010). „Learning To Count Objects in Images.“ In: *Advances in Neural Information Processing Systems 23 (NIPS)*, pp. 1324–1332 (cit. on p. 88).

Lindsay, Bruce G (1988). „Composite Likelihood Methods.“ In: *Contemporary Mathematics* 80 (cit. on pp. 54, 55).

Liu, Dong C. and Jorge Nocedal (1989). „On the Limited Memory BFGS Method for Large Scale Optimization.“ In: *Mathematical Programming* 45, pp. 503–528 (cit. on p. 28).

Liu, Qiang and Alexander Ihler (2011). „Variational Algorithms for Marginal MAP.“ In: *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 453–462 (cit. on pp. 130, 131).

Liu, Tie-Yan (2009). „Learning to Rank for Information Retrieval.“ In: *Foundations and Trends in Information Retrieval* 3.3, pp. 225–331 (cit. on p. 38).

Long, Philip M. and Lei Tan (1998). „PAC Learning Axis-aligned Rectangles with Respect to Product Distributions from Multiple-Instance Examples.“ In: *Machine Learning* 30.1, pp. 7–21 (cit. on p. 39).

Lovasz, László (1983). „Submodular Functions and Convexity.“ In: *Mathematical Programming: The State of the Art*, pp. 235–257 (cit. on p. 49).

Luccio, Flaminia L. (1998). „Almost exact minimum feedback vertex sets in meshes and butter-flies.“ In: *Information Processing Letters*, pp. 59–64 (cit. on p. 129).

MacKay, David J C (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press (cit. on p. 47).

Marroquin, Jose, S. Mitter, and Tomaso Poggio (1987). „Probabilistic Solution of Ill-Posed Problems in Computational Vision.“ In: *Journal of the American Statistical Association* 82.397, pp. 76–89 (cit. on p. 10).

Martins, André L., Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing (2011). „An Augmented Lagrangian Approach to Constrained MAP Inference.“ In: *Proceedings of the 28th International Conference on Machine Learning (ICML)*. Omnipress, pp. 169–176 (cit. on p. 75).

McAllester, David A. and Joseph Keshet (2011). „Generalization Bounds and Consistency for Latent Structural Probit and Ramp Loss.“ In: *Advances in Neural Information Processing Systems 24 (NIPS)*, pp. 2205–2212 (cit. on p. 120).

Mccallum, Andrew, Dayne Freitag, and Fernando Pereira (2000). „Maximum entropy Markov models for information extraction and segmentation.“ In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pp. 591–598 (cit. on pp. 3, 21).

McCormick, S. Thomas (2006). „Submodular function minimization.“ In: *Handbook on Discrete Optimization*. Elsevier, 321—391 (cit. on p. 49).

Meshi, Ofer and Amir Globerson (2011). „An Alternating Direction Method for Dual MAP LP Relaxation.“ In: *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in*

databases - Volume Part II. ECML PKDD'11. Springer-Verlag, pp. 470–483 (cit. on p. 75).

Meshi, Ofer, David Sontag, Tommi Jaakkola, and Amir Globerson (2010). „Learning Efficiently with Approximate Inference via Dual Losses." In: *Proceedings of the 27th International Conference on Machine Learning (ICML)* (cit. on p. 127).

Moussouris, John (1974). „Gibbs and Markov random systems with constraints." In: *Journal of Statistical Physics* 10, pp. 11–33 (cit. on p. 17).

Nesterov, Yurii (1983). „A method of solving a convex programming problem with convergence rate $o(1/k^2)$." In: *Soviet Mathematics Doklady* 27, pp. 372–376 (cit. on pp. 73, 74).

Nesterov, Yurii (2005). „Smooth minimization of non-smooth functions." In: *Mathematical Programming* 103.1, pp. 127–152 (cit. on pp. 74, 111).

Nocedal, Jorge and Stephen J. Wright (1999). *Numerical Optimization*. Springer (cit. on p. 28).

Nowozin, Sebastian and Christoph H. Lampert (2011). „Structured Learning and Prediction in Computer Vision." In: *Foundations and Trends in Computer Graphics and Vision* 6.3-4, pp. 185–365 (cit. on p. 7).

Nowozin, Sebastian, Carsten Rother, Shai Bagon, Toby Sharp, Bangpeng Yao, and Pushmeet Kohli (2011). „Decision tree fields." In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1668–1675 (cit. on pp. 80, 82).

Orlin, James B. (2009). „A faster strongly polynomial time algorithm for submodular function minimization." In: *Mathematical Programming* 118.2, pp. 237–251 (cit. on p. 50).

Papadimitriou, Christos H. and Kenneth Steiglitz (1982). *Combinatorial optimization: algorithms and complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. (cit. on p. 50).

Pearl, Judea (1986). „Fusion, Propagation, and Structuring in Belief Networks." In: *Artificial Intelligence* 29.3, pp. 241–288 (cit. on p. 45).

Pearl, Judea (1990). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann (cit. on pp. 47, 127).

Pletscher, Patrick and Pushmeet Kohli (2012). „Learning low-order models for enforcing high-order statistics." In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 22, pp. 886–894 (cit. on p. vii).

Pletscher, Patrick and Cheng Soon Ong (2012). „Part & Clamp: An efficient algorithm for structured output learning." In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 22, pp. 877–885 (cit. on p. vii).

Pletscher, Patrick, Cheng Soon Ong, and Joachim M. Buhmann (2009). „Spanning Tree Approximations for Conditional Random Fields." In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 5, pp. 408–415 (cit. on p. vii).

Pletscher, Patrick, Cheng Soon Ong, and Joachim M. Buhmann (2010). „Entropy and Margin Maximization for Structured Output Learning." In: *Proceedings of the 20th European Conference on Machine Learning (ECML)*. Vol. 6321. Lecture Notes in Computer Science, pp. 83–98 (cit. on p. vii).

Pletscher, Patrick and Sharon Wulff (2011). „A Combined LP and QP Relaxation for MAP." In: *NIPS Workshop on Discrete Optimization in Machine Learning (DISCML)* (cit. on p. vii).

Pletscher, Patrick and Sharon Wulff (2012). „LPQP for MAP: Putting LP Solvers to Better Use." In: *Proceedings of the 29th International Conference on Machine Learning (ICML)* (cit. on p. vii).

Pletscher, Patrick, Sebastian Nowozin, Pushmeet Kohli, and Carsten Rother (2011). „Putting MAP back on the Map." In: *33rd Annual Symposium of the German Association for Pattern Recognition (DAGM)*. Vol. 6835. Lecture Notes in Computer Science. Springer, pp. 111–121 (cit. on p. vii).

Potetz, Brian (2007). „Efficient Belief Propagation for Vision Using Linear Constraint Nodes." In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 18, 83).

Potts, Renfrey B. (1952). „Some generalized order-disorder transformations." In: *Mathematical Proceedings of the Cambridge Philosophical Society* 48.1, pp. 106–109 (cit. on p. 16).

Quattoni, Ariadna, Sy Bor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell (2007). „Hidden Conditional Random Fields." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.10, pp. 1848–1852 (cit. on pp. 29, 112).

Rabiner, Lawrence R. (1989). „A tutorial on hidden Markov models and selected applications in speech recognition." In: *Proceedings of IEEE*. Vol. 77, pp. 257–286 (cit. on p. 45).

Rabinovich, Andrew, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie (2007). „Objects in context.“ In: *IEEE 11th International Conference on Computer Vision (ICCV)*, pp. 1–8 (cit. on p. 36).

Ranjbar, Mani, Tian Lan, Yang Wang, Steven N. Robinovitch, Ze-Nian Li, and Greg Mori (2012). „Optimizing Non-Decomposable Loss Functions in Structured Prediction.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. to appear (cit. on p. 88).

Ravikumar, Pradeep, Alexh Agarwal, and Martin J Wainwright (2010). „Message-passing for Graph-structured Linear Programs: Proximal Methods and Rounding Schemes.“ In: *Journal of Machine Learning Research* 11, pp. 1043–1080 (cit. on p. 77).

Ravikumar, Pradeep and John Lafferty (2006). „Quadratic Programming Relaxations for Metric Labeling and Markov Random Field MAP Estimation.“ In: *Proceedings of the 23rd international conference on Machine learning (ICML)*, pp. 737–744 (cit. on pp. 61, 69).

Ray, Soumya and Mark Craven (2005). „Supervised Versus Multiple Instance Learning: An Empirical Comparison.“ In: *Proceedings of the 22nd International Conference on Machine Learning (ICML)*. ACM Press, pp. 697–704 (cit. on p. 119).

Robbins, Herbert and Sutton Monro (1951). „A stochastic approximation method.“ In: *Annals of Mathematical Statistics* 22, pp. 400–407 (cit. on pp. 29, 55, 133).

Robert, Christian P. (2001). *The Bayesian Choice. From decision Theoretic Foundations to Computational Implementation*. Springer (cit. on p. 10).

Robert, Christian P. and George Casella (2005). *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Secaucus, NJ, USA: Springer-Verlag New York, Inc. (cit. on p. 52).

Roth, Dan (1996). „On the hardness of approximate reasoning.“ In: *Artificial Intelligence* 82, pp. 273–302 (cit. on p. 43).

Roth, Stefan and Michael J Black (2009). „Fields of Experts.“ In: *International Journal of Computer Vision* 82.2, pp. 205–229 (cit. on p. 83).

Rother, Carsten, Pushmeet Kohli, Wei Feng, and Jiaya Jia (2009). „Minimizing sparse higher order energy functions of discrete variables.“ In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1382–1389 (cit. on pp. 18, 89).

Savchynskyy, Bogdan, Jörg H Kappes, Stfan Schmidt, and Christoph Schnörr (2011). „A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling.“ In: *The 24th IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR), pp. 1817–1823 (cit. on pp. 73, 74).

Schlesinger, M I (1976). „Syntactic analysis of two-dimensional visual signals in noisy conditions." In: *Kibernetika* 4, pp. 113–130 (cit. on p. 59).

Schmidt, Uwe, Qi Gao, and Stefan Roth (2010). „A Generative Perspective on MRFs in Low-Level Vision." In: *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1751–1758 (cit. on p. 55).

Schraudolph, Nicol N. and Dmitry Kamenetsky (2009). „Efficient Exact Inference in Planar Ising Models." In: *Advances in Neural Information Processing Systems 22 (NIPS)*, pp. 1417–1424 (cit. on p. 138).

Shewchuk, Jonathan R (1994). *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep. Pittsburgh, PA, USA (cit. on p. 29).

Shotton, Jamie, John Winn, Carsten Rother, and Antonio Criminisi (2009). „TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context." In: *International Journal of Computer Vision* 81.1, pp. 2–23 (cit. on pp. 36, 37).

Snow, Dan, Paul Viola, and Ramin Zabih (2000). „Exact Voxel Occupancy with Graph Cuts (470)." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 345–352 (cit. on p. 49).

Sontag, David, Talya Meltzer, Amir Globerson, Yair Weiss, and Tommi Jaakkola (2008). „Tightening LP Relaxations for MAP using Message-Passing." In: *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI)*, pp. 503–510 (cit. on pp. 60, 74, 76, 80).

Sontag, David, Ofer Meshi, Tommi Jaakkola, and Amir Globerson (2010). „More data means less inference: A pseudo-max approach to structured learning." In: *Advances in Neural Information Processing Systems 23 (NIPS)*, pp. 2181–2189 (cit. on pp. 54, 133).

Sriperumbudur, Bharath and Gert Lanckriet (2009). „On the Convergence of the Concave-Convex Procedure." In: *Advances in Neural Information Processing Systems 22 (NIPS)*, pp. 1759–1767 (cit. on p. 75).

Stobbe, Peter and Andreas Krause (2010). „Efficient Minimization of Decomposable Submodular Functions." In: *Advances in Neural Information Processing Systems 23 (NIPS)* (cit. on pp. 18, 50, 91).

Sudderth, Erik B. and Michael Jordan (2008). „Shared Segmentation of Natural Scenes Using Dependent Pitman-Yor Processes." In: *Ad-*

vances in Neural Information Processing Systems 21 (NIPS), pp. 1585–1592 (cit. on p. 83).

Sutskever, Ilya and Tijmen Tieleman (2010). „On the Convergence Properties of Contrastive Divergence." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 9, pp. 789–795 (cit. on p. 55).

Sutton, Charles and Andrew Mccallum (2012). „An Introduction to Conditional Random Fields." In: *Foundations and Trends in Machine Learning* 4.4, pp. 267–373 (cit. on p. 19).

Szeliski, Richard, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother (2008). „A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.6, pp. 1068–1080 (cit. on p. 83).

Szummer, Martin, Pushmeet Kohli, and Derek Hoiem (2008). „Learning CRFs Using Graph Cuts." In: *10th European Conference on Computer Vision (ECCV)*, pp. 582–595 (cit. on p. 83).

Tarlow, Daniel, Inmar E. Givoni, and Richard S. Zemel (2010). „HOP-MAP: Efficient Message Passing with High Order Potentials." In: pp. 812–819 (cit. on p. 18).

Tarlow, Daniel and Richard Zemel (2011). „Big and Tall: Large Margin Learning with High Order Losses." In: *CVPR 2011 Workshop on Inference in Graphical Models with Structured Potentials* (cit. on pp. 88, 94).

Tarlow, Daniel and Richard S. Zemel (2012). „Structured Output Learning with High Order Loss Functions." In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP 22, pp. 1212–1220 (cit. on pp. 88, 94).

Taskar, Ben, Carlos Guestrin, and Daphne Koller (2003). „Max-Margin Markov Networks." In: *Advances in Neural Information Processing Systems 16 (NIPS)* (cit. on pp. 3, 30, 117, 118).

Tsochantaridis, Ioannis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun (2005). „Large Margin Methods for Structured and Interdependent Output Variables." In: *Journal of Machine Learning Research* 6, pp. 1453–1484 (cit. on pp. 3, 30, 102, 103, 132).

Tsoumakas, Grigorios and Ioannis Katakis (2007). „Multi-Label Classification: An Overview." In: *International Journal on Data Warehousing and Mining* 3.3, pp. 1–13 (cit. on p. 35).

Valiant, Leslie G. (1979). „The Complexity of Computing the Permanent.“ In: *Theoretical Computer Science* 8, pp. 189–201 (cit. on pp. 13, 43).

Vandenberghe, Lieven (2012). *Fast proximal gradient methods*. Lecture notes (cit. on p. 73).

Vapnik, Vladimir N. (1995). *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer New York Inc. (cit. on p. 25).

Vazirani, Vijay V. (2001). *Approximation Algorithms*. Springer (cit. on p. 128).

Vickrey, Davd, Cliff Chiung-Yu Lin, and Daphne Koller (2010). „Non-Local Contrastive Objectives.“ In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pp. 1103–1110 (cit. on pp. 56, 126, 127).

Viterbi, Andrew (1967). „Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.“ In: *IEEE Transactions on Information Theory* 13.2, pp. 260–269 (cit. on p. 45).

Wainwright, Martin J., Tommi Jaakkola, and Alan S. Willsky (2005a). „A new class of upper bounds on the log partition function.“ In: *IEEE Transactions on Information Theory*. Vol. 51. 7, pp. 2313–2335 (cit. on p. 127).

Wainwright, Martin J, Tommi Jaakkola, and Alan S Willsky (2005b). „MAP estimation via agreement on trees: message-passing and linear programming.“ In: *IEEE Transactions on Information Theory* 51.11, pp. 3697–3717 (cit. on p. 70).

Wainwright, Martin J. and Michael I. Jordan (2008). „Graphical Models, Exponential Families, and Variational Inference.“ In: *Foundations and Trends in Machine Learning* 1.1-2, pp. 1–305 (cit. on pp. 23, 24, 44, 49, 59, 60, 64, 66).

Winkler, Gerhard (2006). *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction*. Stochastic Modelling and Applied Probability. Secaucus, NJ, USA: Springer-Verlag (cit. on p. 17).

Woodford, Oliver J, Carsten Rother, and Vladimir Kolmogorov (2009). „A Global Perspective on MAP Inference for Low-Level Vision.“ In: *IEEE 12th International Conference on Computer Vision (ICCV)*, pp. 2319–2326 (cit. on pp. 83, 89).

Xu, Jun, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma (2008). „Directly optimizing evaluation measures in learning to rank.“ In: *Proceedings of the 31th annual international ACM SIGIR conference on*

Research and development in information retrieval, pp. 107–114 (cit. on p. 38).

Yanover, Chen, Talya Meltzer, and Yair Weiss (2006). „Linear Programming Relaxations and Belief Propagation – An Empirical Study.“ In: *Journal of Machine Learning Research* 7, pp. 1887–1907 (cit. on p. 79).

Yedidia, Jonathan S., William T. Freeman, and Yair Weiss (2005). „Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms.“ In: *IEEE Transactions on Information Theory* 51, pp. 2282–2312 (cit. on pp. 47, 70).

Yu, Chun-Nam and Thorsten Joachims (2009). „Learning Structural SVMs with Latent Variables.“ In: *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pp. 1169–1176 (cit. on pp. 32, 112).

Yuille, Alan L. (2002). „CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation.“ In: *Neural Computation* 14.7, pp. 1691–1722 (cit. on p. 65).

Yuille, Alan L (2004). „The convergence of contrastive divergences.“ In: *Advances in Neural Information Processing Systems 17 (NIPS)* (cit. on p. 55).

Yuille, Alan L and Anand Rangarajan (2003). „The concave-convex procedure.“ In: *Neural Computation* 15.4, pp. 915–936 (cit. on pp. 65, 121).

Zhang, Tong (2005). „Class-size Independent Generalization Analysis of Some Discriminative Multi-Category Classification.“ In: *Advances in Neural Information Processing Systems 17 (NIPS)* (cit. on p. 114).

Zhang, Tong and Frank J. Oles (2000). „Text Categorization Based on Regularized Linear Classification Methods.“ In: *Information Retrieval* 4, pp. 5–31 (cit. on pp. 106, 114).

NOTATION

PROBABILITY

SYMBOL	MEANING
$P(\cdot)$	probability mass function
Z	partition sum; normalization constant of a distribution
A	logarithm of the partition sum
A^*	conjugate dual of the log partition sum
$\mathbb{E}_P[X]$	expected value of X, w.r.t. distribution P
$\mathrm{Cov}_P[X]$	covariance of X, w.r.t. distribution P
$D_{KL}(q\|p)$	Kullback-Leibler divergence of distributions q and p
$I(p,q)$	mutual information of p and q
$H(P)$	entropy of the distribution P
β	inverse temperature in a Gibbs distribution
$\mathbb{I}_{\texttt{expr}}(x)$	indicator function returning 1 if the Boolean expression `expr` involving variable x is true and 0 otherwise.

STRUCTURED OUTPUT

SYMBOL	MEANING
$\mathcal{Y}_i$	domain of the i-th output variable
y_i	individual output variable, $y_i \in \mathcal{Y}_i$
$\mathcal{Y}$	product space of all the individual output domains
$\boldsymbol{y}$	all the output variables of an example
$\mathcal{X}$	domain of the input variables
$\boldsymbol{x}$	all the input variables of an example
$\mathcal{Z}_i$	domain of the i-th hidden output variable
z_i	individual hidden output variable, $z_i \in \mathcal{Z}_i$
$\mathcal{Z}$	product space of all individual hidden output domains
$\boldsymbol{z}$	all the hidden output variables of an example
$\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$	feature map of an example $(\boldsymbol{x}, \boldsymbol{y})$
$\boldsymbol{\psi}(\boldsymbol{y}'\|\boldsymbol{x}, \boldsymbol{y})$	feature map difference: $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}') - \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$
$\Delta_{\boldsymbol{y}^*}(\boldsymbol{y})$	loss when predicting $\boldsymbol{y}$ instead of $\boldsymbol{y}^*$
$\boldsymbol{w}$	(linear) parameters of a structured model
$\ell(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$	surrogate loss of parameter $\boldsymbol{w}$ for example $(\boldsymbol{x}, \boldsymbol{y})$

FACTOR GRAPHS AND GRAPHICAL MODELS

SYMBOL	MEANING
$E(\boldsymbol{y})$	energy of output configuration $\boldsymbol{y}$
$\theta_c(\boldsymbol{y}_c)$	potential of factor c and assignment $\boldsymbol{y}_c$
$\bar{\theta}_c(\boldsymbol{y}_c)$	negative potential (score) of factor c and assignment $\boldsymbol{y}_c$
$\mathcal{G}$	graph
$\mathcal{V}$	vertices of a graph
$\mathcal{E}$	edges of a graph
$\mathcal{N}(i)$	neighboring vertices of vertex i
d_i	degree of the i-th vertex
$\mathcal{FG}$	factor graph
$\boldsymbol{\mu}$	marginal variables
$\mathcal{M}$	marginal polytope
$\mathcal{L}_{\mathcal{G}}$	local marginal polytope for graph $\mathcal{G}$

ACRONYMS

MAP	Maximum-A-Posteriori
MPE	Most Probable Explanation
MMSE	Minimum Mean Squared Error
MPM	Maximum Posteriori Marginal
ERM	Empirical Risk Minimization
MRF	Markov Random Field
CRF	Conditional Random Field
HCRF	Hidden Conditional Random Field
SVM	Support Vector Machine
mi-SVM	microscopic Multiple-Instance SVM
MI-SVM	macroscopic Multiple-Instance SVM
LP	Linear Program
QP	Quadratic Program
L-BFGS	limited-memory Broyden-Fletcher-Goldfarb-Shanno
OWL-QN	Orthant-Wise Limited-memory Quasi-Newton
SGD	Stochastic Gradient Descent
FISTA	Fast Iterative Shrinkage-Thresholding
ADMM	Alternating Direction Method of Multipliers
CCCP	Concave-Convex Procedure
DC	Difference of convex functions
SA	Simulated Annealing

MCMC	Markov Chain Monte Carlo
CD	Contrastive Divergence
KL	Kullback-Leibler
BP	Belief Propagation
MPLP	Max-Product Linear Programming
TRWS	Sequential Tree-Reweighted Message Passing
LPQP	Linear and Quadratic Program relaxation
LPQP-U	LPQP with Uniform Penalty
LPQP-T	LPQP with Tree-weighted Penalty
IBFS	Incremental Breadth First Search
FVS	Feedback Vertex Set
HOG	Histogram of Oriented Gradients
SIFT	Scale-invariant feature transform
OCR	Optical Character Recognition
VOC	Visual Object Classes

INDEX

COLOPHON

This document was typeset in LaTeX using the typographical look-and-feel `classicthesis`. Most of the graphics in this thesis are generated using `pgfplots` and `pgf/tikz`. The bibliography is typeset using `biblatex`.